AD-A088 570 PENNSYLVANIA STATE UNIV UNIVERSITY PARK DEPT OF INDU--ETC F/G 9/2
ADVANCES IN MULTICRITERIA ENGINEERING DESIGN: COMPUTATIONAL TEC--ETC(U)
JUL 80 T L ELCHAK, J P IGNIZIO, T YANG N00014-79-C-0730
UNCLASSIFIED NL

1 OF 2
AD A088570

LEVEL II

3

ADVANCES IN MULTICRITERIA ENGINEERING DESIGN: Computational Techniques

FINAL REPORT

U.S. Navy Contract: N00014-79-C-0730

"Multiobjective Engineering Design via Nonlinear Goal Programming"

Prepared for:

The Office of Naval Research
Department of the Navy
800 N. Quincy Street
Arlington, Virginia 22217

by

Terrence L. Michak Taeyong Yang
Dr. James P. Ignizio (Principal Investigator)

July 1980

THE PENNSYLVANIA STATE UNIVERSITY
College of Engineering
Department of Industrial and Management Systems Engineering
207 Hammond Building
University Park, Pa. 16802

(6) Advances in Multicriteria Engineering Design:

Computational Techniques

(3)

(9) FINAL REPORT

(15) U.S. Navy Contract: N00014-79-C-0730

"Multiobjective Engineering Design via Nonlinear Goal Programming"

Prepared for:

The Office of Naval Research
Department of the Navy
800 N. Quincy Street
Arlington, Virginia 22217

by:

(10) Terrence L. Elchak

James P. Ignizio
(Principal Investigator)

and

Taeyong Yang

at the

Department of Industrial & Management Systems Engineering
207 Hammond Building
The Pennsylvania State University
University Park, Pennsylvania 16802

131

(11) July 1980

410768

ABSTRACT

The procedure for solving large-scale, nonlinear multiobjective optimization problems requires the use of a computer coded algorithm which is both computationally efficient and mathematically valid. The multicriteria methodology, Goal Programming, has proven to be an effective technique for the solution of such problems.

The problem of this study was the synthesis of the goal programming methodology into an efficient computer algorithm which could be used by the problem solver to solve a general class of nonlinear, multiobjective (NLGP) engineering design problems. Three such codes were developed into a computer package. Each code is capable of solving an NLGP problem. Each code has modifications which make it more attractive than the others for application to particular types of NLGP problems. The results show that these codes are effective tools for solving nonlinear goal programming design problems.

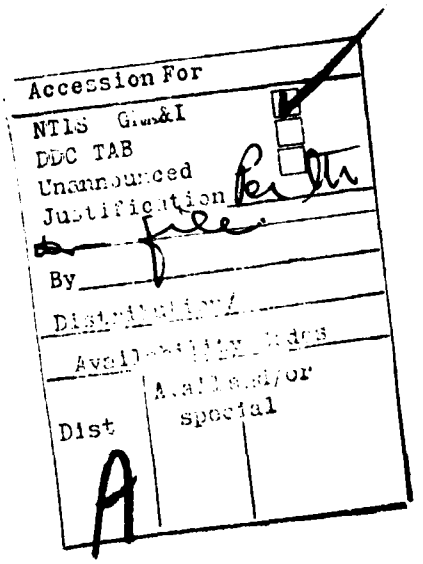


TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Problem Statement

The design of the majority of large-scale, complex engineering systems requires the specification of parameters which will best achieve a system of multiple and conflicting design objectives. These objectives may be linear and nonlinear in nature. Real systems can often contain hundreds and often thousands of specific design objectives. The design parameters which optimize the performance of these objectives can number in the thousands. Unfortunately, traditional methods of modeling and optimizing such systems are restricted to problems encompassing a single design goal. Computational techniques for these methods are quite often further restricted to either small problems or uncomplicated objectives, or both.

A satisfactory technique for the solution of large-scale engineering design problems with multiple, conflicting objectives is contained within the multicriteria tool of goal programming [18]. This computational technique has been applied to a wide variety of problems. The computer code presented in [18] is intended for very small problems, while those in [2] and [14] have been adapted to meet the design requirements of very specific problems. There is a need for an improved nonlinear goal programming (NLGP) computer code which can be used without modification to solve a wide variety of general large-scale nonlinear, multiobjective design problems. The purpose of this study is to develop such a computer code.

1.2 Specific Objectives

The specific objectives of this study are:

1. To develop a computer code (or codes) which can solve NLGP problems with up to thousands of objectives and thousands of variables,

2. To test and validate the code(s) for computational efficiency,
3. To supply a User's Manual describing in detail the use of the code(s).

1.3 Summary of Results

The research work resulted in the development of three computer codes. All three are similar in technique. Each code has minor modifications from the others which makes it more suitable for application to particular NLGP problems.

The codes were tested on a variety of problems varying in the complexity of the nonlinear objectives and problem size (the number of variables and objectives). The results obtained indicate that each code is computationally correct and yielding satisfactory results.

This report contains information about the development of the computer codes with details about their structure and performance. A User's Manual has been written which details specifically how to use the three computer codes.

1.4 Overview of Following Chapters

An analysis of the results of this research are contained in the following chapters.

Chapter II contains a background sketch of the general nonlinear goal programming model. It lists advantages and limitations of the technique in solving design problems. The third chapter then provides a description of the computer codes, their limitations and modifications. Some examples are presented in Chapter IV, along with the empirical results from the testing of the codes. Finally, some conclusions about this study and recommendations for further research are provided in Chapter V.

CHAPTER II

THE MATHEMATICAL PROGRAMMING PROBLEM

2.1 The General Nonlinear Goal Programming Model

The primary emphasis of this research has been directed towards the area of multicriteria decision analysis, and in particular, nonlinear goal programming. For this reason, a discussion of the considerable number of other mathematical programming optimization techniques will not be presented. The entire focus of discussion will be on multicriteria techniques.

The multicriteria technique called Goal Programming was first presented by Charnes and Cooper [1] in the 1950's. Ignizio [18] later expanded the method to solve nonlinear as well as linear models. More recently, this technique has found a broad range of applications to solving design problems in many areas [3,13,14,15,16,17,19,20,21,22,23,24].

The general nonlinear goal programming model can be described by the following [18].

A set of mathematical equations are established by the analyst or design engineer which describe the problem as functions of the design parameters. These design equations are the objectives, some or all of which may be nonlinear in nature. The design parameters are the decision variables. Associated with the objectives are desired aspiration levels. They represent the performance of the system at the optimal design parameters.

Define the following:

(1) $\{f_1(\bar{x}), f_2(\bar{x}), \ldots, f_i(\bar{x})\}$ is the set of objectives

(2) $\bar{x} = (x_1, x_2, \ldots, x_J)$ is the vector of decision variables

(3) $\{b_1, b_2, \ldots, b_i\}$ is the set of aspiration levels for the objectives.

Then,

$$f_i(\bar{x}) \quad \begin{matrix} \geq \\ = \\ \leq \end{matrix} \quad b_i, \qquad i = 1, 2, \ldots, m$$

represent the objectives or goals for the system.

It is desired to find the values of the decision variables such that the objectives meet the specified aspiration levels. Since this is not always possible due to the conflicting nature of many objectives, deviation variables are added to the design objectives. These variables measure the difference between the desired and the actual aspiration levels.

Then,

$$f_i(\bar{x}) + n_i - p_i = b_i, \qquad i = 1, 2, \ldots, m$$

are the design objectives where n_i measures the negative deviation from the aspired level for objective i, and p_i measures the positive deviation for objective i.

A major assumption underlying many versions of goal programming is that the analyst can establish preemptive priorities for each objective or group of objectives. This implies that the objectives can be ordered such that achievement of the objectives at any one priority level is immeasurably preferred to the achievement of the objectives of any lower level.

Any objectives which are absolute (i.e., they must be satisfied) are assigned to priority level (P_1). The remaining objectives are then grouped and assigned priority levels (i.e., P_2, P_3, ..., P_j).

The goal programming algorithm first satisfies, as nearly as possible, the objectives with the highest priority level. It then proceeds to satisfy the objectives of the next priority level, as nearly as is possible, without degrading the achievement of any objective in a higher priority level. This process continues until all priority levels have been considered.

The satisfaction of these various priority levels is measured by an achievement vector, $\bar{a}$. This vector has one component for each priority level. Each component, a_m, is a weighted linear function of the deviation variables which are contained in the objectives of that priority level m.

Let $\{g_1(\bar{n},\bar{p}), g_2(\bar{n},\bar{p}), \ldots, g_k(\bar{n},\bar{p})\}$ represent the set of these linear functions, one for each priority level. Each function can be represented as:

$$g_k(\bar{n},\bar{p}) = \sum_{i \in P_k} (\mu_i n_i + w_i p_i),$$

the weighted sum of negative and positive deviation variables for all the objectives contained in priority level P_k, where μ_i and w_i are the weights.

Since both negative and positive deviation variables are required for each objective, minimization of these variables will attempt to achieve the aspiration levels of the objectives. The achievement vector $\bar{a}$ is structured as an ordered set such that the preemptive priority structure is maintained.

The standard form of the nonlinear goal programming model is as follows:

Find $\bar{x} = (x_1, x_2, \ldots, x_j)$

to minimize $\bar{a} = \{g_1(\bar{n},\bar{p}), g_2(\bar{n},\bar{p}), \ldots, g_k(\bar{n},\bar{p})\}$

subject to

$$f_i(\bar{x}) + n_i - p_i = b_i, \qquad i = 1, 2, \ldots, m$$

are satisfied, $\bar{n}$ and $\bar{p}$ are both nonnegative,

(1) $g_k(\bar{n},\bar{p})$ is a weighted linear function of the deviation variables

(2) k is the priority level associated with the objective

(3) the dimension of $\bar{a}$ represents the number of preemptive priority levels, (k),

and

(4) the number of preemptive priorities are equal to or less than the number of objectives.

The value of $\bar{a}$ will be equal to the zero vector if all the objectives meet their aspiration levels. The value of a_k will be a positive value if one or more objectives in priority level k are not met. The goal programming solution is considered optimal if it is the lexicographic minimum (i.e., the value of $\bar{a}$ is lexicographically the same as or less than the value of $\bar{a}$ for any other feasible solution). If a_1 is non-zero, an absolute objective cannot be met, and the solution is considered unimplementable.

To further clarify the general nonlinear goal programming model, an example problem is presented next.

2.2 A Target Allocation Example

The example problem selected to demonstrate the formulation of an NLGP model is a target allocation problem which was previously formulated and solved by Rice, Bracken, and Pennington [28] using a single objective optimization technique, the sequential unconstrained minimization technique [5,6].

This problem examines the allocation of weapon systems among targets with emphasis being placed on maximizing the amount of damage inflicted on enemy targets. The weapon systems in this problem are limited to two types of attack aircraft originating from two aircraft carriers, one conventional and one nuclear. Additionally, the number of each aircraft is considered fixed. This creates a scenario in which the primary objective of the problem is to allocate a fixed inventory of aircraft among carriers so as to maximize total damage on the enemy targets.

The following parameters and variables are introduced for this problem:

x_{ij} = the number of aircraft of type j assigned to carrier i, i = 1,2 and j = 1,2

a_{jk} = the fraction of target k (either in numbers of specific targets or target area) left undamaged following a mission by an aircraft of type j, j = 1,2 and k = 1,2, ..., 5

y_{ijk} = the number of missions flown against target k by an aircraft of type j from carrier i, i = 1,2, j = 1,2 and k = 1,2, ..., 5

Additional parameters are defined for the objectives with regards to the fixed supply of aircraft and carriers:

b_j = total number of available aircraft of type j

c_j = the carrier space required by each aircraft of type j

d_i = total space available for plane storage on carrier i

f_{ijk} = the fraction of total missions available in the planning period used in flying one sortie against target k by an aircraft of type j from carrier i.

The objectives for this problem are formulated as follows:

Aircraft availability constraints

$$\sum_{i=1}^{2} x_{ij} \leq b_j \qquad \text{for } j = 1,2$$

Carrier space constraints

$$\sum_{j=1}^{2} c_j x_{ij} \leq d_i \qquad \text{for } i = 1,2$$

Limitations on missions due to aircraft availability

$$x_{ij} - \sum_{k=1}^{5} f_{ijk}\, y_{ijk} \geq 0 \qquad \text{for } \begin{matrix} i = 1,2 \\ j = 1,2 \end{matrix}$$

Damage inflicted on targets

The total damage inflicted is the sum of damages to all targets. The damage to any individual target may be expressed in terms of the parameters, a_{jk}. The damage inflicted on target k by missions of all aircraft from all carriers may be formulated as:

$$1 - \prod_{i=1}^{2} \prod_{j=1}^{2} a_{jk}^{\,y_{ijk}} .$$

The total damage inflicted to all targets is the sum of damages to each target given above. Each target can be given a military target value, μ_k, which weights that target in terms of its military importance. Military importance could be an estimate of value in terms of specific numbers (i.e., number of planes destroyed), a target area or fraction of area, dollars, or some other measure.

The damage function then is:

$$\sum_{k=1}^{5} \mu_k \left(1 - \prod_{i=1}^{2} \prod_{j=1}^{2} \alpha_{jk}^{y_{ijk}}\right).$$

Also note that the variable x_{ij} and y_{ijk} are restricted to non-negative values.

The problem to be solved is: given the parameters for the problem, determine the values of the variables x_{ij} and y_{ijk} such that the total damage function is maximized subject to the aircraft, carrier, and mission constraints.

This problem will now be converted into a standard NLGP formulation. The problem variables must be converted into a standard format for the NLGP model. These model variables are listed in Table 2.1. The parameters necessary to formulate the model objectives are listed in Tables 2.2 and 2.3 and Figure 2.1. Negative and positive deviation variables have been added to the objectives.

Aircraft availability objectives

Goal 1: $x_1 + x_3 + n_1 - p_1 = 27$

Goal 2: $x_2 + x_4 + n_2 - p_2 = 102$

Carrier space objectives

Goal 3: $2920\, x_1 + 1770\, x_2 + n_3 - p_3 = 112{,}300$

Goal 4: $2920\, x_3 + 1770\, x_4 + n_4 - p_4 = 147{,}100$

Limitations on missions due to aircraft availability

Goal 5: $x_1 - 0.06452\,(x_5 + x_7 + x_9) - 0.06250\,(x_6 + x_8) + n_5 - p_5 = 0$

Goal 6: $x_2 - 0.05556\,(x_{10} + x_{12} + x_{14}) - 0.05264\,(x_{11} + x_{13}) + n_6 - p_6 = 0$

$$\text{Goal 7:} \quad x_3 - 0.06896\, x_{15} = 0.06452\,(x_{16} + x_{19}) - 0.06250\,(x_{17} + x_{19}) + n_7 - p_7 = 0$$

$$\text{Goal 8:} \quad x_4 - 0.05882\, x_{20} - 0.05556\,(x_{21} + x_{24}) - 0.05264\,(x_{22} + x_{23}) + n_8 - p_8 = 0$$

Damage objective

$$\text{Goal 9:} \quad 40\,(1 - 0.99978^{(x_5 + x_{15})}\, 0.99953^{(x_{10} + x_{20})}) + 10\,(1 - 0.99978^{(x_6 + x_{16})}\, 0.99953^{(x_{11} + x_{21})}) + 50\,(1 - 0.99978^{(x_7 + x_{17})}\, 0.99953^{(x_{12} + x_{22})}) + n_9 - p_9 = 0$$

Non-negativity objectives

$\bar{x}, \bar{n}, \bar{p} \geq \bar{0}$ (the zero vector).

Note that only those terms with non-zero u_k values appear in the damage objective. Scenario 1 data were used.

All that remains to be done is to formulate the achievement vector. This problem has two priority levels. Goals one through eight compose priority level one. These are absolute objectives (i.e., limited resources) which cannot be violated. Goals one through four are resource objectives. These cannot be exceeded. Therefore, the positive deviation variables for these objectives ($p_1 - p_4$) are minimized at priority level one. This prohibits overachievement of these objectives. Goals five through eight relate aircraft to missions, where the total number sent on any mission can equal to or less than the total supply but can never exceed it. Hence, the negative deviation variables ($n_5 - n_8$) are minimized. This prohibits underachievement of these objectives.

Table 2.1 Target Allocation Problem Variables

Description		Problem Variable	NLGP Model Variable
Distribution of aircraft of type j to carrier i		y_{11}	x_1
		y_{12}	x_2
		y_{21}	x_3
		y_{22}	x_4
Missions flown from carrier 1 to target k	Using aircraft of type 1	y_{111}	x_5
		y_{112}	x_6
		y_{113}	x_7
		y_{114}	x_8
		y_{115}	x_9
	Using aircraft of type 2	y_{121}	x_{10}
		y_{122}	x_{11}
		y_{123}	x_{12}
		y_{124}	x_{13}
		y_{125}	x_{14}
Missions flown from carrier 2 to target k	Using aircraft of type 1	y_{211}	x_{15}
		y_{212}	x_{16}
		y_{213}	x_{17}
		y_{214}	x_{18}
		y_{215}	x_{19}
	Using aircraft of type 2	y_{221}	x_{20}
		y_{222}	x_{21}
		y_{223}	x_{22}
		y_{224}	x_{23}
		y_{225}	x_{24}

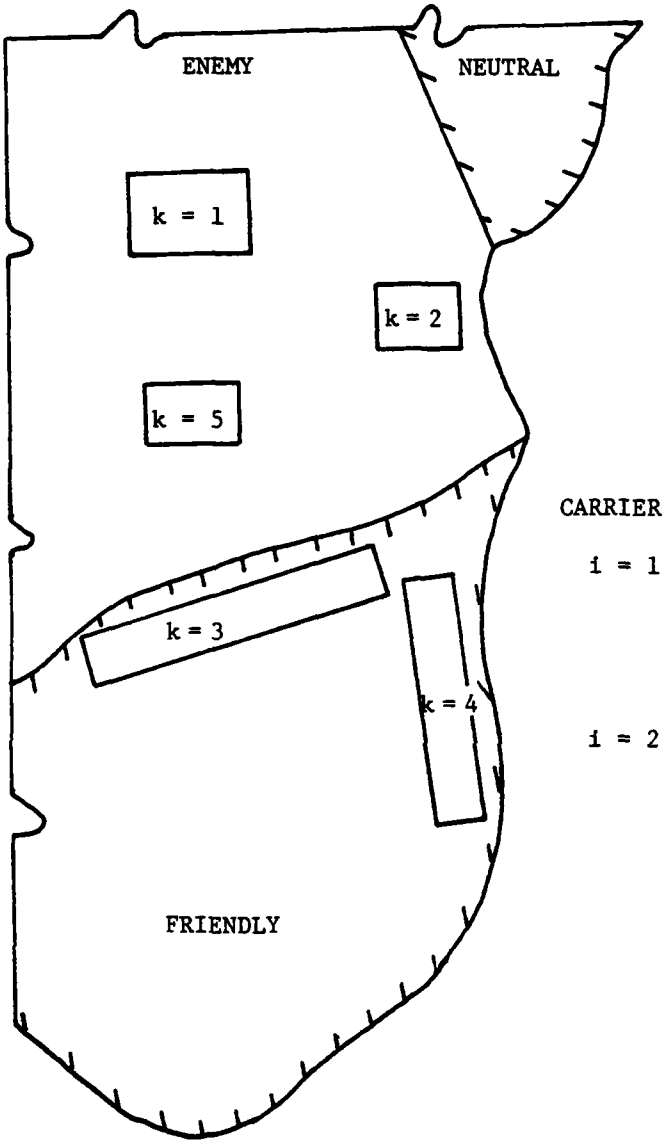
Table 2.2 Target Allocation Problem Parameters [28]

Type	Value for Model	
a_{jk}	$a_{11} = 0.99978$	$a_{21} = 0.99953$
	$a_{12} = 0.99978$	$a_{22} = 0.99953$
	$a_{13} = 0.99978$	$a_{23} = 0.99953$
	$a_{14} = 0.99935$	$a_{24} = 0.99960$
	$a_{15} = 0.99935$	$a_{25} = 0.99960$
b_j	$b_1 = 27$	(A-6 Aircraft)
	$b_2 = 102$	(A-7 Aircraft)
c_j	$c_1 = 2,920$ sq. ft.	(A-6 Aircraft)
	$c_2 = 1,770$ sq. ft.	(A-7 Aircraft)
d_j	$d_1 = 112,300$ sq. ft.	(CVA)
	$d_2 = 147,100$ sq. ft.	(CVAN)
f_{ijk}	$f_{111} = 0.06452$	$f_{211} = 0.06896$
	$f_{112} = 0.06250$	$f_{212} = 0.06452$
	$f_{113} = 0.06452$	$f_{213} = 0.06250$
	$f_{114} = 0.06250$	$f_{214} = 0.06250$
	$f_{115} = 0.06452$	$f_{215} = 0.06452$
	$f_{121} = 0.05556$	$f_{221} = 0.05882$
	$f_{122} = 0.05264$	$f_{222} = 0.05556$
	$f_{123} = 0.05556$	$f_{223} = 0.05264$
	$f_{124} = 0.05264$	$f_{224} = 0.05264$
	$f_{125} = 0.05556$	$f_{225} = 0.05556$

Note: f_{ijk} values were determined by distance to the target and required maintenance between missions.

Table 2.3 Military Target Values [28]

Target Set	Scenario 1	Scenario 2	Scenario 3
1. Airfields	40	45	0
2. Fuel Storage Depots	10	15	0
3. Close Support	50	40	45
4. Short Interdiction	0	0	45
5. Command and Control	0	0	10



CARRIER	TARGET	DISTANCE
i = 1	k = 1	400
	k = 2	200
	k = 3	400
	k = 4	300
	k = 5	400
i = 2	k = 1	500
	k = 2	400
	k = 3	300
	k = 4	300
	k = 5	400

Figure 2.1 Sample scenario for target allocation problem [28]

The second priority level contains only one objective, the damage objective. The aspiration level for Goal 9 is set equal to zero, and since this objective is maximized, underachievement of the objective is minimized by minimizing the negative deviation variable (n_9). Any overachievement will then be indicated by a non-zero value for p_9, the measure of total damage inflicted on all targets.

The NLGP model for this problem is:

Find the values of $\bar{x} = (x_1, x_2, \ldots, x_{24})$

so as to

minimize $\bar{a} = (p_1 + p_2 + p_3 + p_4 + n_5 + n_6 + n_7 + n_8, n_9)$

subject to

Goal 1 through Goal 9

and $\bar{x}, \bar{n}, \bar{p} \geq \bar{0}$.

The discussion of the solution to this target allocation problem will be deferred until later chapters.

2.3 Advantages and Limitations

There is still no single best method for solving nonlinear models, whether they contain a single objective or multiple objectives. Each method has advantages and disadvantages inherent to that method. Some advantages and limitations of the NLGP method are reviewed here [18].

The greatest advantage of the NLGP method over conventional methods is that this method allows the analyst or design engineer to model the problem realistically. Most problems have more than a single objective, and generally, some of these objectives will be conflicting in nature. The objectives of the system do not have to be massaged to fit into some prescribed mold so that a solution can be obtained.

A major advantage of the NLGP method over conventional methods is that a solution to the problem is always obtained. This is due to the direct search technique used in the method and the ordered set of preemptive priority levels. Rather than considering infeasibility, solutions are either implementable (i.e., $a_1 = 0$) or unimplementable (i.e., $a_1 > 0$). The achievement vector indicates the degree to which the desired aspiration levels are met.

In general in nonlinear optimization, there is no way to guarantee that the solution obtained will be a global optimum. The NLGP technique performs no better or worse than other methods in this respect. In actual practice, this technique has proved to be a robust method for the solution of some very complex problems.

The priority structure does have limitations, however. The existence of a preemptive priority structure may not always be a valid assumption. Absolute objectives (i.e., natural laws, limited resources) cannot be violated and belong in the highest priority level. The number and arrangement of objectives in lower priority levels may not be as obvious and can depend upon the analyst, or engineer's understanding and insight into the problem. Quite often some objectives must be switched between priority levels or entire levels rearranged to obtain satisfactory results, especially when little is known apriori about the problem.

The use of weighting factors in the achievement vector is an advantage because these factors enable the analyst or design engineer to weight objectives only within a single priority level. The proper weighting of objectives is not a clearcut procedure, however, and these weights often must be established through an iterative process.

Finally, aspiration levels for some objectives may not always be obvious. Again, proper determination of these levels may have to be accomplished through a number of trials and, in any case, evaluated in the final result via a sensitivity analysis.

CHAPTER III

COMPUTATIONAL TECHNIQUES

3.1 Introduction

A variety of methods can be employed to solve nonlinear models [5,6,7,8,9,10,11,12,27,30]. The success of a method over other methods to find a solution may be relative to the particular problem as well as to initial problem parameters. A straightforward approach for solving large-scale nonlinear models is one employing direct search techniques. The method selected for the nonlinear goal programming technique is based on an extension of the search method of Hooke and Jeeves [12].

The Hooke-Jeeves method is an accelerated pattern search technique. It is based on the assumption that any set of moves which has been successful in optimizing the objectives is worth repeating. The algorithm starts with an initial point (i.e., an n-dimensional vector), perturbs each variable one at a time and tests for an improved solution. When a search has been successful, future increments for the pattern search moves are increased. When a search is unsuccessful in finding an improved solution, future increments are decreased. This pattern search technique has been shown to be a simple and effective method for solving actual problems with large numbers of variables, large number of objectives, and a variety of convergence criteria [12]. It is a simple method which makes it computationally efficient. This makes it attractive for use with the NLGP algorithm.

3.2 The Nonlinear Goal Programming Computer Code

The software package developed under this contract contains three separate computer codes. All three codes can be used to solve an NLGP problem. All three codes are very similar in technique. Each code has some minor modifications which makes it better suited to solve a different variation of NLGP problem. The basic computer code and its operating characteristics will be discussed in detail in this section. The remaining two codes will be discussed in the section covering modifications. All program versions are written in American National Standard FORTRAN IV.

The basic computer code is referred to as the nonlinear goal programming/modified pattern search - ridge search code, NLGP/MPS-RS code. This is Version 1 of the code. A complete listing of this code is contained in Appendix A.

The NLGP/MPS-RS: Version 1 code is composed of a small main program and seven subprograms. Two of these seven subprograms must be supplied in part by the user for each new NLGP problem. Figure 3.1 is a schematic diagram showing the operating relationships of the subprograms.

The MAIN program contains data initialization statements, type declaration statements, and other specification statements that are necessary for the definition of the program parameters and variables. The MAIN program calls subroutine DATAIN and subroutine HJALG, the Hooke-Jeeves pattern search algorithm. Subroutine DATAIN is used to input all problem data. Once this is accomplished, all further control of the program proceeds through subroutine HJALG. Table 3.1 gives a brief description of the modules making up the computer code. Module length is given as compiled on an IBM 370/3033 System using the FORTRAN WATFIV compiler.

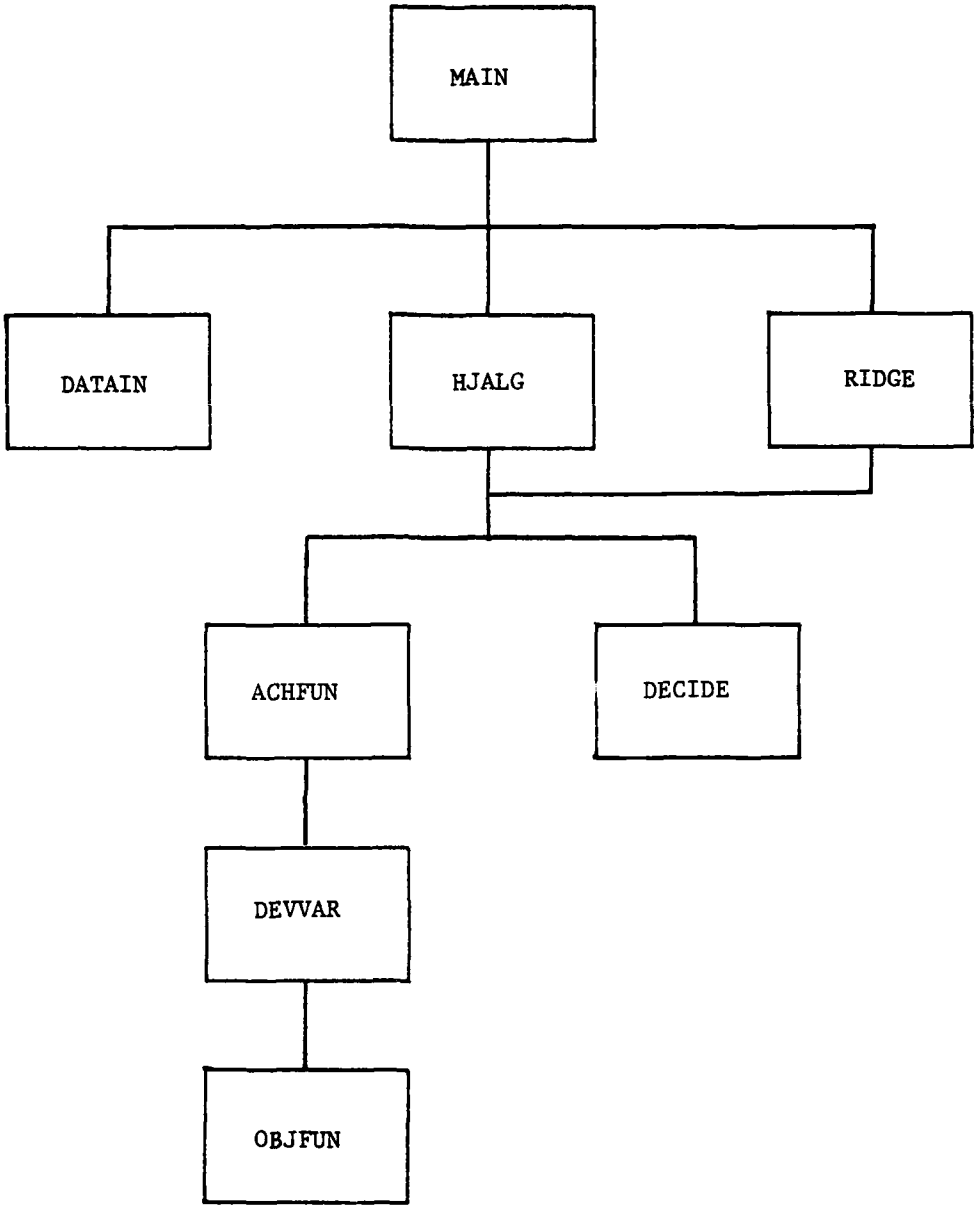


Figure 3.1 Program Relationship Diagram - NLGP/MPS-RS Code: Versions 1 and 2

Table 3.1 Program Module Information NLGP/MPS-RS Code: Versions 1 and 2

Program Module	Module Length No. of bytes (in decimal)	Description	Calls	Called By
MAIN Program	632	Serves as executive program for the pattern search	DATAIN HJALG	------
Subroutine DATAIN	3344	Reads into storage all problem data	------	MAIN
Subroutine HJALG	6048	Performs pattern search through exploratory and pattern moves	ACHFUN DECIDE RIDGE	MAIN
Subroutine RIDGE	4248	Performs a resolution ridge search when the pattern search no longer moves	ACHFUN DECIDE	HJALG
Subroutine DECIDE	2000	Compares achievement vectors and tests for termination	------	HJALG RIDGE
Subroutine ACHFUN	Problem Dependent Minimum of 368	Computes the values of the achievement vector	DEVVAR	HJALG RIDGE
Subroutine DEVVAR	680	Computes the values of the positive and negative deviation variables	OBJFUN	ACHFUN
Subroutine OBJFUN	Problem Dependent Minimum of 304	Computes the values of the objective functions	------	DEVVAR
Total:	Minimum of 17624 bytes			

Each subprogram of the code will be discussed in terms of its operation and the program parameters and variables which it uses. The complete FORTRAN language listing of each subprogram is contained in Appendix A. A list of all program variables and parameters is given in Table 3.2 A complete guide on how to select the values of program parameters is contained in the User's Manual.

The computer program is presently dimensioned to handle problems with up to 2500 objectives, 2500 variables, and 10 priority levels. These dimensions can be altered to increase the problem size which can be solved.

Subroutine DATAIN

Subroutine DATAIN is used to input all the necessary data for the NLGP problem. Once this data is read into internal storage, an echo check of this data is printed as a means of checking it. Table 3.3 contains a listing of the data which must be input and its form.

Variables and parameters used in DATAIN:

NOBJ
NPRIOR
NVAR
NMAX
EPS
ACCEL
REDUCE
DELTA
X
LBD
UBD
RHS
EPSY.

Table 3.2 Computer Code Variables and Parameters
NLGP/MPS-RS Code: Version 1

Name	Description
NOBJ	Number of problem objectives
NPRIOR	Number of priority levels in the achievement vector
NVAR	Number of decision variables
NMAX	Maximum number of pattern search cycles allowed before termination; set by user
NCYCLE	Counter of pattern search cycles
X	Decision variable vector for perturbations in pattern search
XBASE	Decision variable base point vector for pattern search
A	Achievement vector at a test point
ABEST	Achievement vector for best solution during pattern search
APOS, ANEG	Temporary achievement vectors for the resolution ridge search
EPS, IPS	Perturbation step-size scalars used for exploratory moves
LBD, UBD	Lower and upper bound vectors on decision variables
OBJ	Objective function vector
RHS	Desired aspiration levels for objectives
N, P	Negative and positive deviation vectors
EPSY	Test vector for termination of algorithm by meeting specified tolerances
ACCEL	Acceleration factor for pattern moves
REDUCE	Step-size reduction factor
DELTA	Minimum allowable step-size for exploratory moves
IMPROV	A test switch; if IMPROV = 1, X improves the solution for ABEST, if IMPROV = 0, X does not improve this solution

Table 3.2 Continued

Name	Description
IPRINT	An output switch; if IPRINT = 1, complete results are printed out at every pattern search cycle, if IPRINT = 0, only the final results are printed.
ITERM	A termination testing indicator; if ITERM = 1, the solution for A is compared to ABEST and then tested for termination against EPSY, if ITERM = 0, only a comparison of A and ABEST is made.
IPROG	A counter for number of improvements, if any, during the exploratory moves
IACCEL	An indicator for pattern moves; if IACCEL = 1, the pattern moves was successful, if IACCEL = 0, no pattern move was attempted, if IACCEL = -1, the pattern move was unsuccessful
SAVE	A temporary storage variable for pattern and ridge moves
I	Counter for all do loops containing the objective functions, OBJ
J	Counter for all do loops containing the priority levels of the achievement vector, A and ABEST
K	Counter for all do loops containing the decision variables, X and XBASE

Table 3.3 Data Input Guide NLGP/MPS-RS
Code: Version 1

Card	Data	Type	Format
1	NOBJ	Integer *4	I5
	NPRIOR	"	I5
	NVAR	"	I5
	NMAX	"	I5
	IPRINT	Integer *2	I5
2 - $(\frac{NVAR}{8} - 1)$	X	Real *4	8F10.0
3 - $(\frac{NVAR}{8} - 1)$	LBD	Real *4	8F10.0
4 - $(\frac{NVAR}{8} - 1)$	UBD	Real *4	8F10.0
5 - $(\frac{NOBJ}{8} - 1)$	RHS	Real *4	8F10.0
6 - $(\frac{NPRIOR}{8} - 1)$	EPSY	Real *4	8F10.0
7	EPS	Real *4	F10.0
	ACCEL	"	F10.0
	REDUCE	"	F10.0
	DELTA	"	F10.0

Subroutine HJALG

Subroutine HJALG is the Hooke-Jeeves pattern search Algorithm as modified to solve NLGP problems. Figure 3.2 is a schematic diagram showing the flow through the pattern search algorithm. A detailed computer logic flow diagram is contained in Figure 3.3.

Variables and parameters used in HJALG:

NOBJ	IPRINT
NPRIOR	ITERM
NVAR	IPROG
NMAX	IACCEL
NCYCLE	IPS
X	EPS
XBASE	LBD
A	UBD
ABEST	IMPROV
SAVE	
ACCEL	
REDUCE	
DELTA	

Subroutine RIDGE

Subroutine RIDGE is used to calculate a new temporary head point when the exploratory moves of the pattern search fail to find any improved points near the base point, XBASE. RIDGE is a heuristic technique which uses information for each set of pairwise perturbations, positive and negative around a variable x(k). It attempts to move in a direction which is oblique to the original axes of exploration, and therefore, move away from the present base point around which the pattern has collapsed.

The basis for RIDGE comes from Rosenbrock's technique for rotation of coordinates [29] and Mugele's "poor man's optimizer" [26]. RIDGE requires much less storage of information than Rosenbrock's method. It is constructed specifically to use information from the achievement vector to establish a direction for continued search. It is called everytime the step size, IPS,

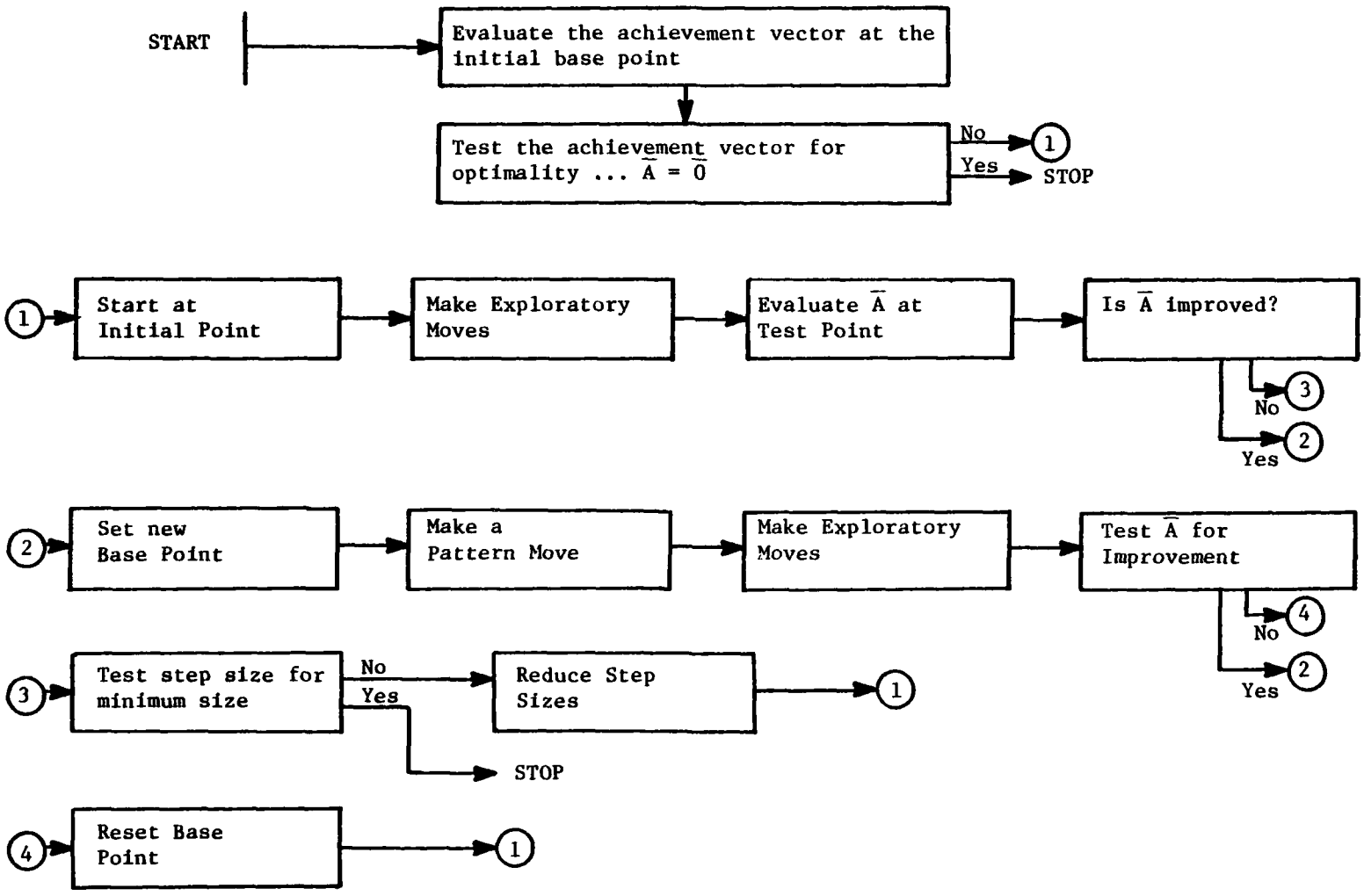


Figure 3.2 Pattern Search Algorithm Flow Diagram

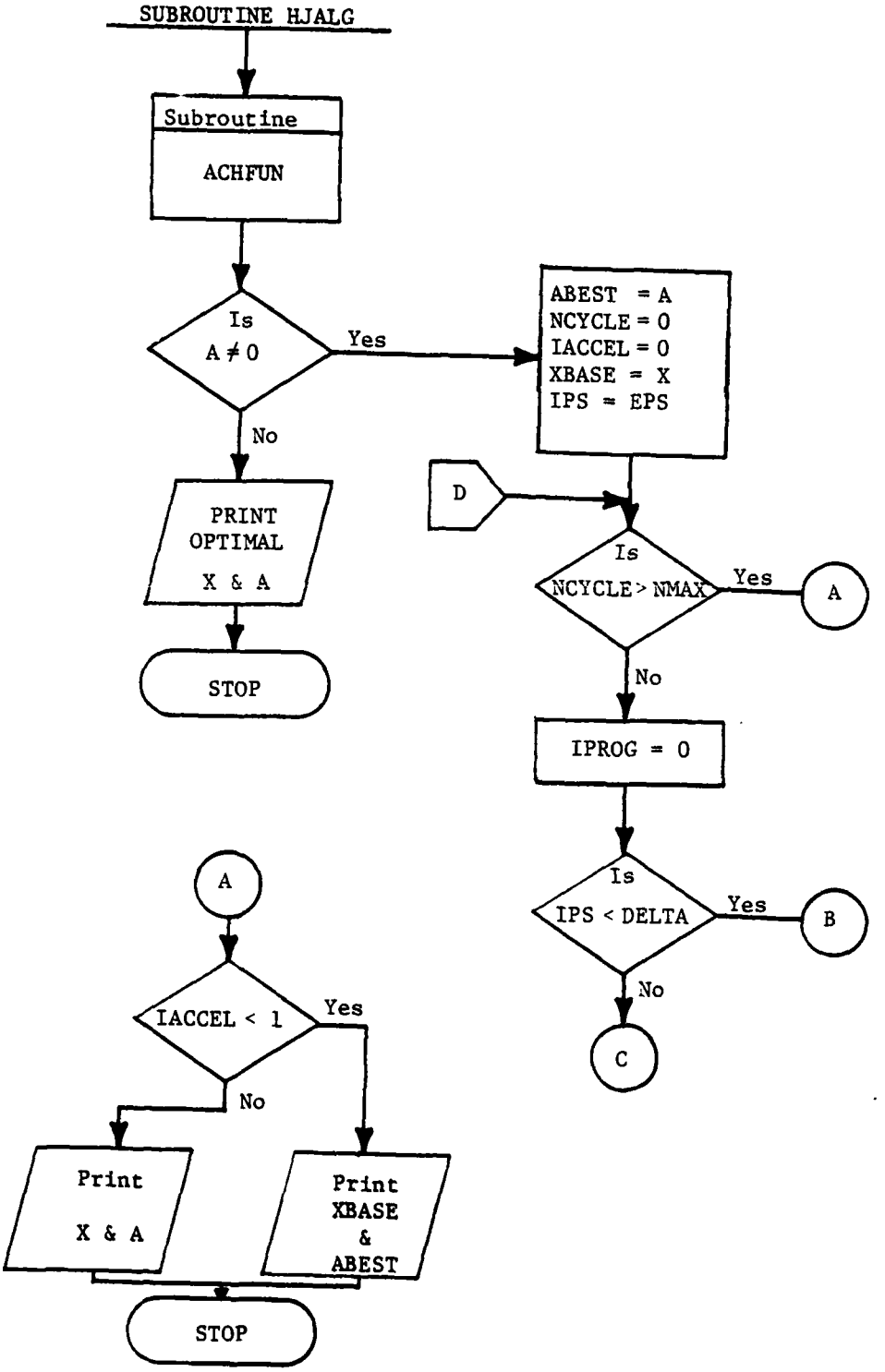


Figure 3.3 Flow Diagram for Subroutine HJALG

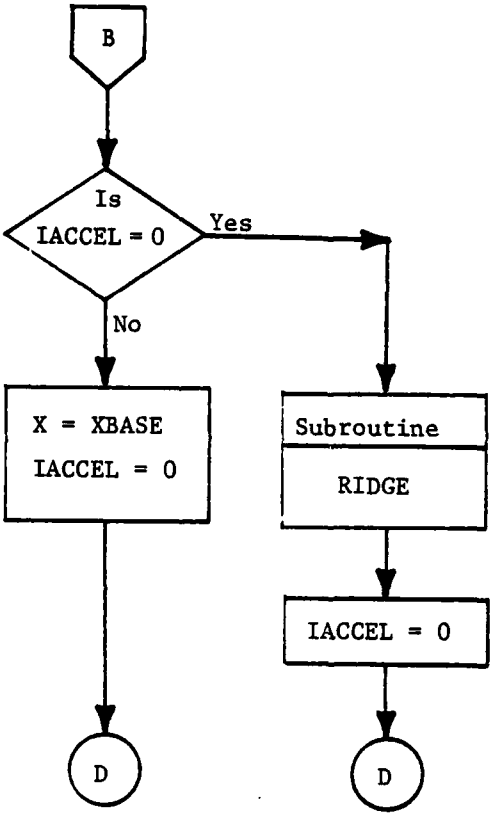


Figure 3.3 Continued

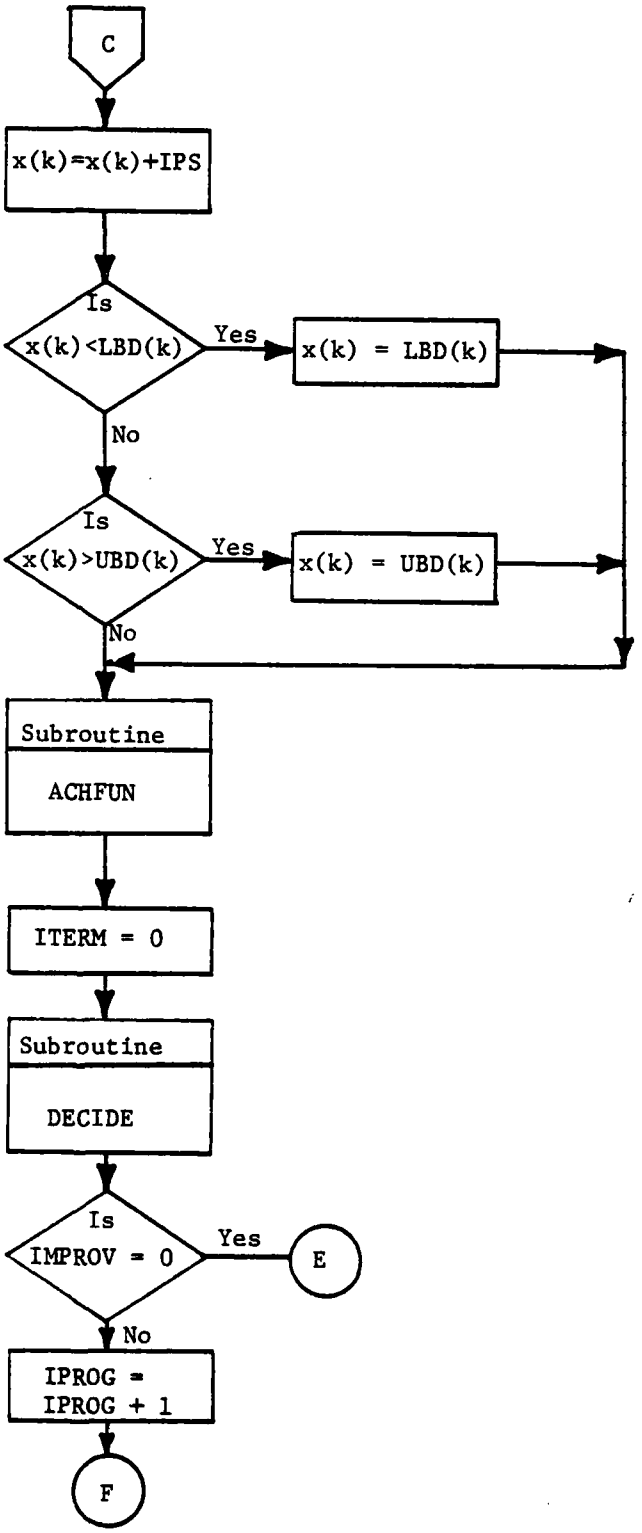


Figure 3.3 Continued

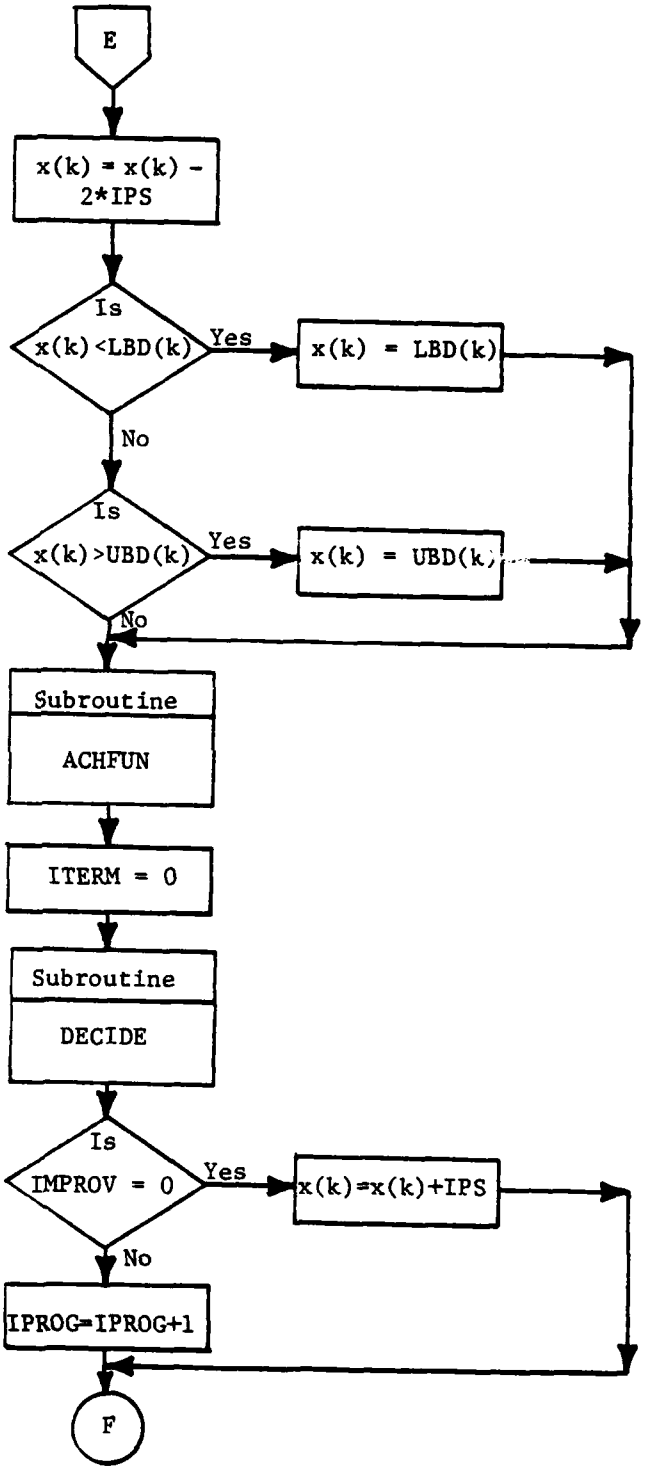


Figure 3.3 Continued

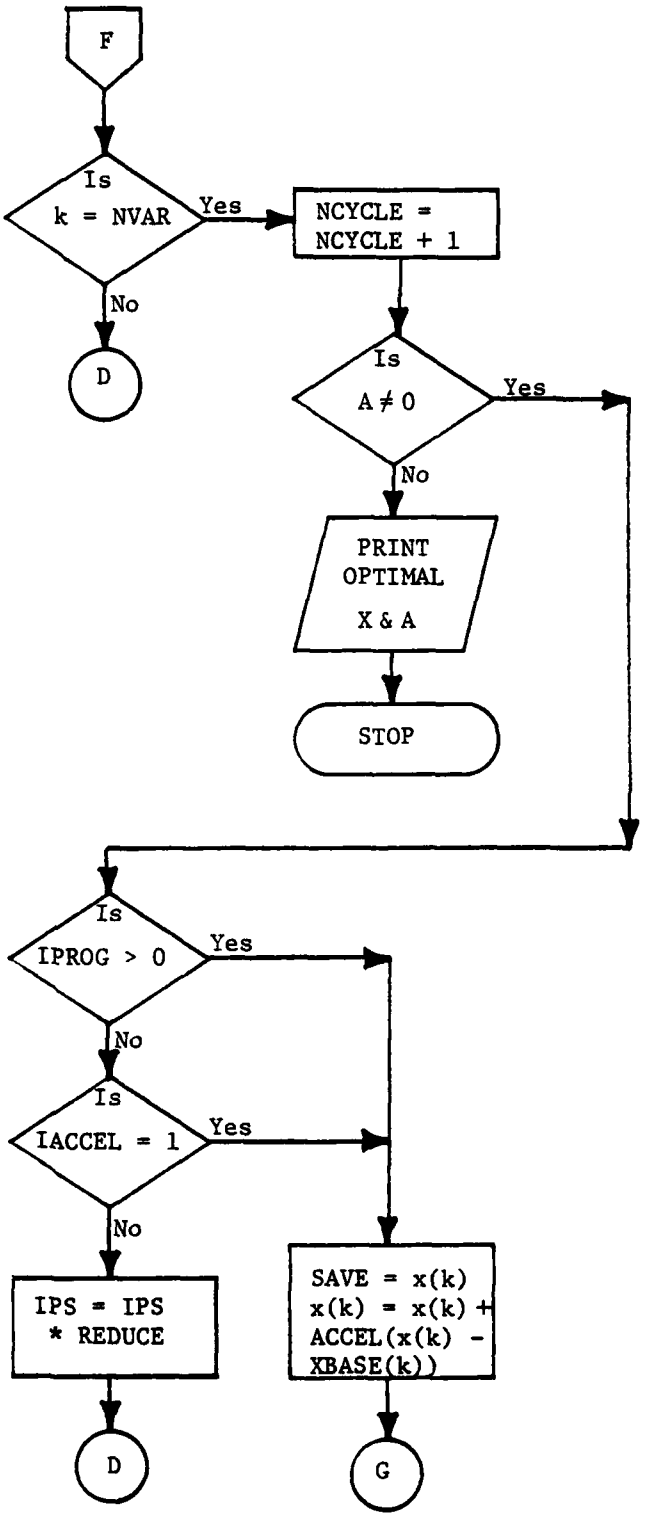


Figure 3.3 Continued

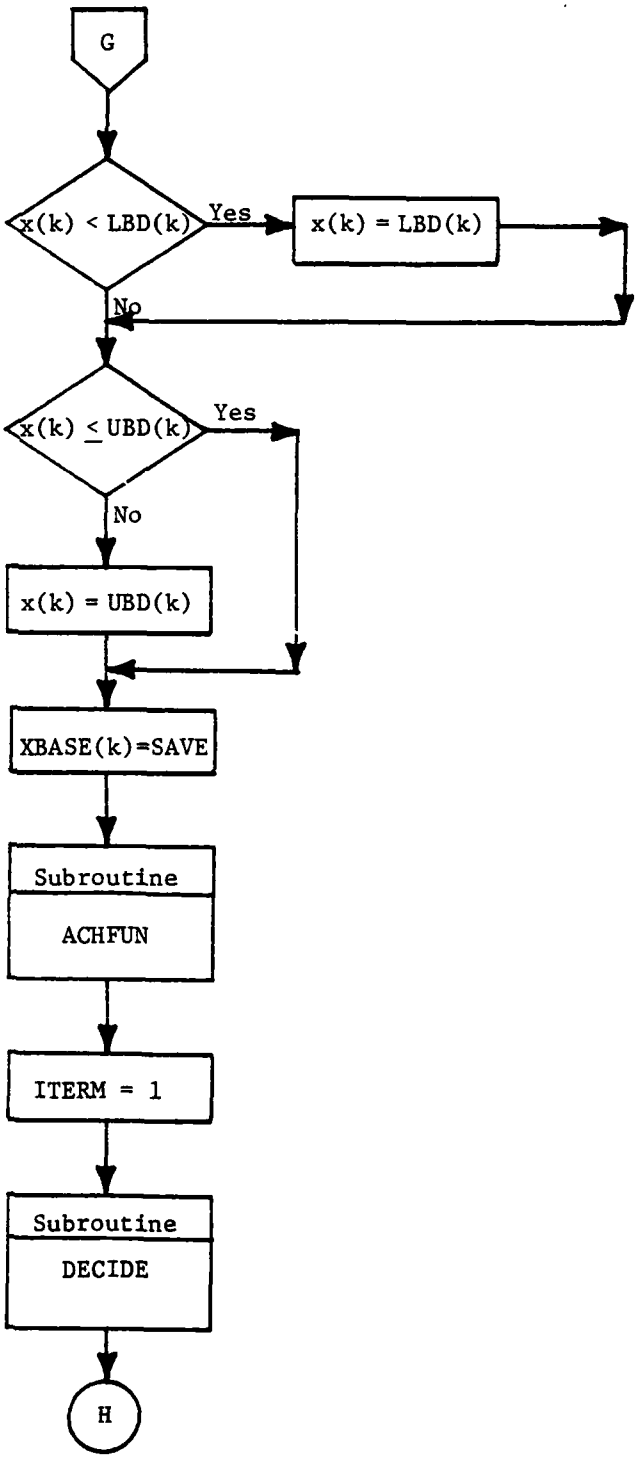


Figure 3.3 Continued

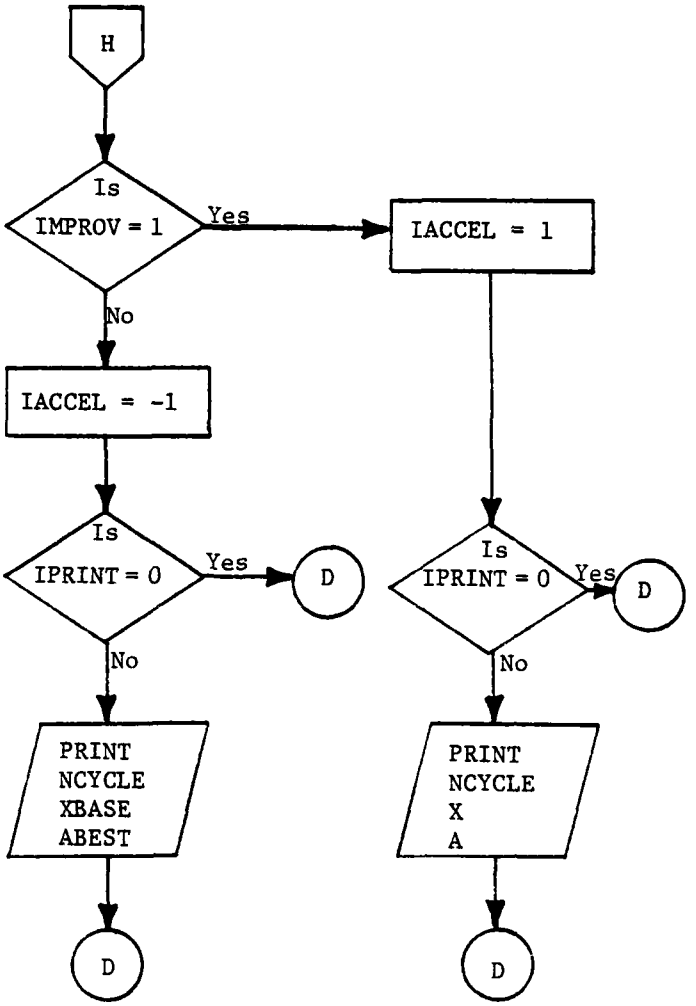


Figure 3.3 Continued

becomes smaller in value than the minimum allowable step size, DELTA.

Figure 3.4 shows the flow diagram for subroutine RIDGE.

Variables and parameters used in RIDGE:

X	NPRIOR
XBASE	SAVE
IPS	NVAR
EPS	ITERM
LBD	IMPROV
UBD	ACCEL
APOS	
ANEG	
A	
ABEST	

Subroutine DECIDE

Subroutine DECIDE has two main purposes: (1) To compare the achievement vector at a test point, A, against the achievement vector for the best point, ABEST. If A is preferred over ABEST, it becomes ABEST and the corresponding x vector, X, becomes XBASE, the new base point. If A is not preferred over ABEST, the program continues with no changes to the variables. and (2) To test the achievement vector for termination of the algorithm. Termination can occur by two methods: first, all elements of the achievement vector have zero values; this is an optimal solution, and secondly, when all elements of the achievement vector have values less than the specified tolerance vector, EPSY. This is the test by tolerances. Figure 3.5 is the flow diagram for subroutine DECIDE.

Variables and parameters used in DECIDE:

A
ABEST
NPRIOR
IMPROV
ITERM
X
EPSY

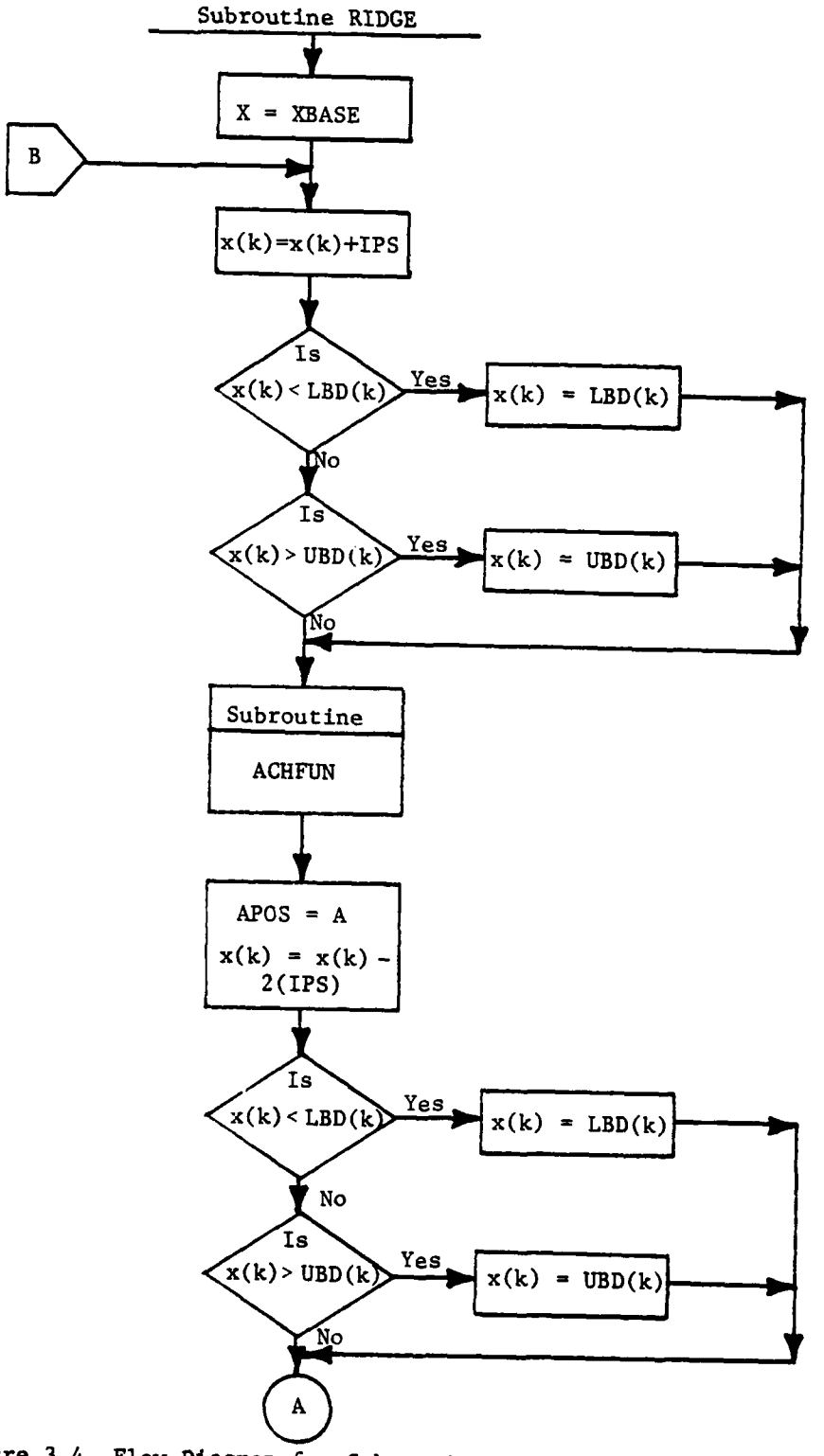


Figure 3.4 Flow Diagram for Subroutine Ridge

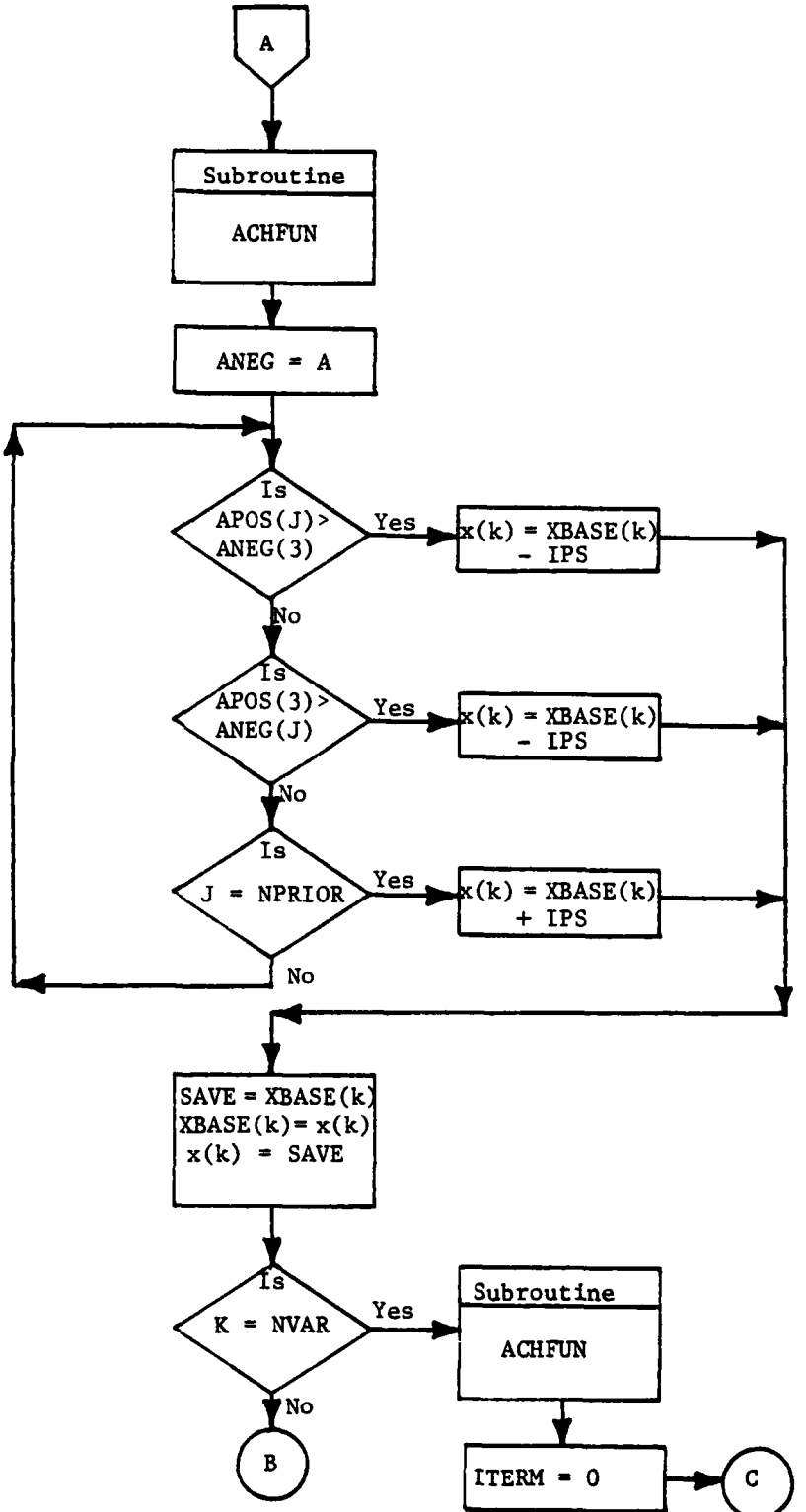


Figure 3.4 Continued

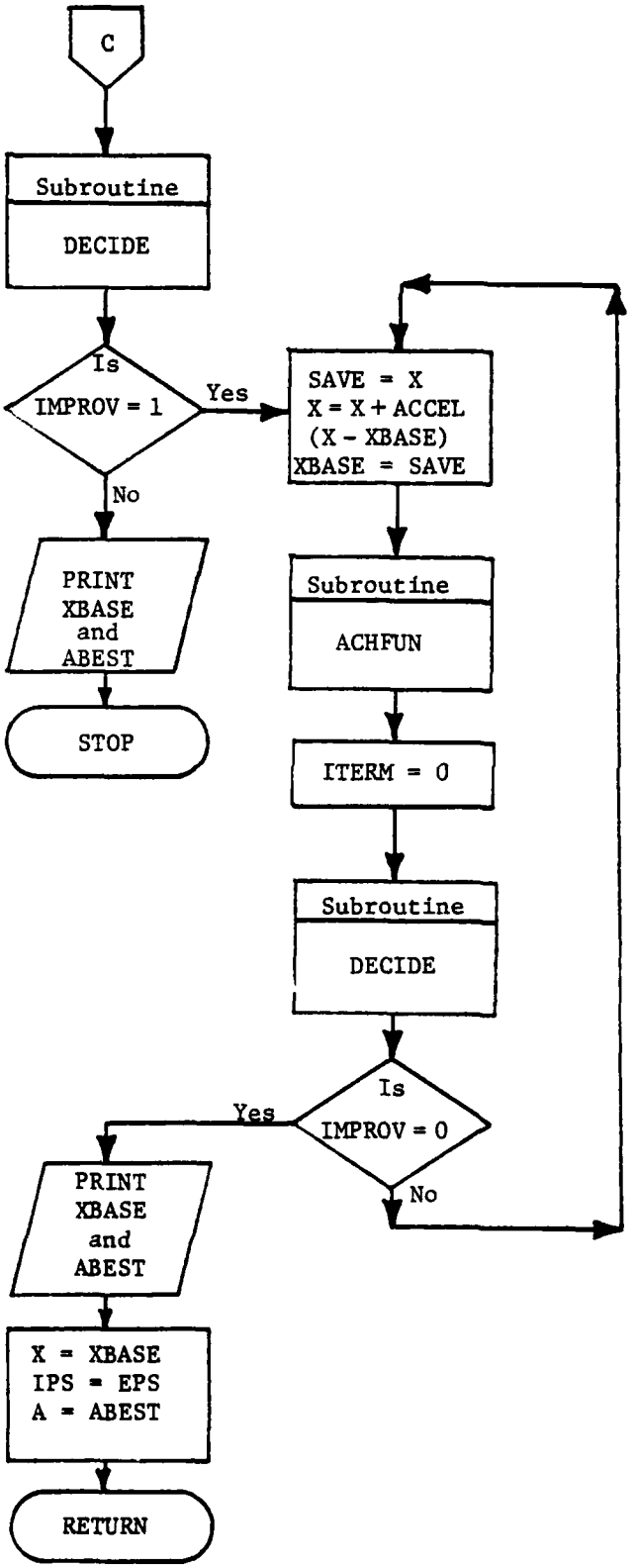


Figure 3.4 Continued

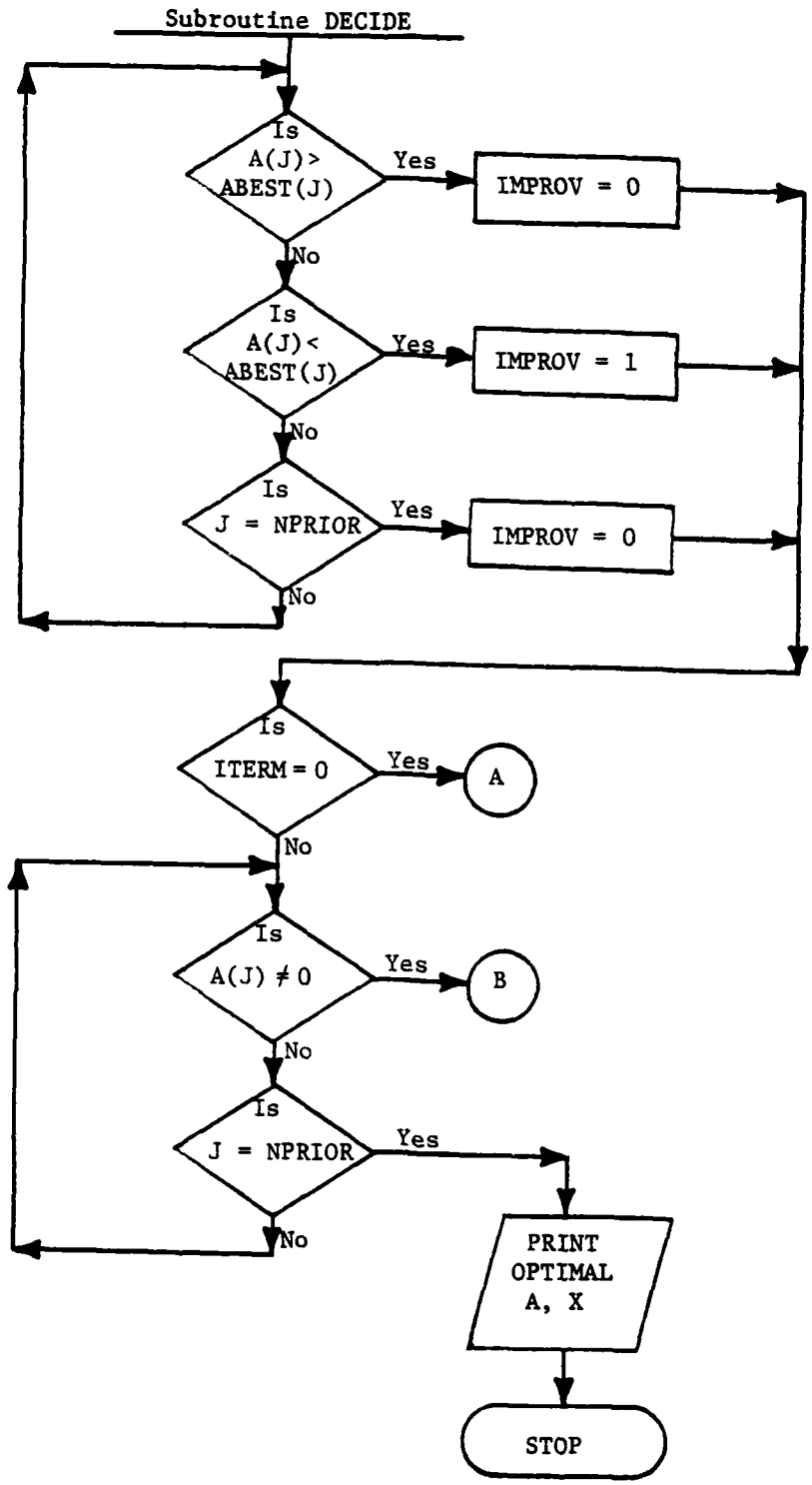


Figure 3.5 Flow Diagram for Subroutine DECIDE

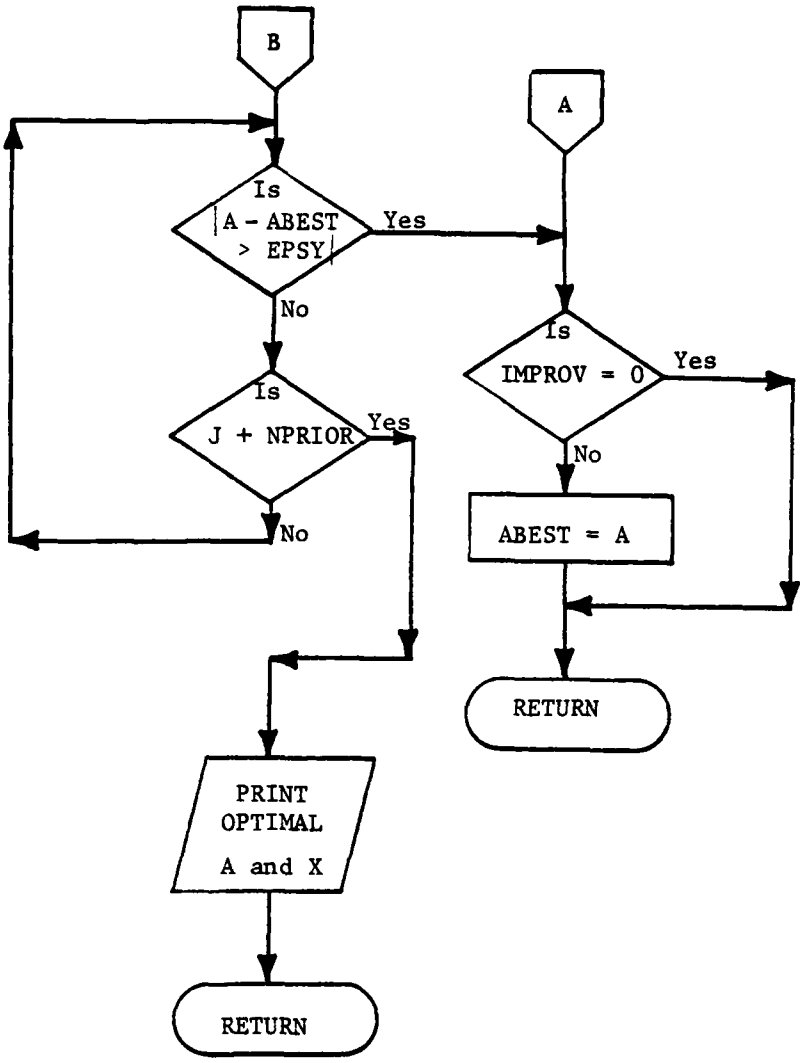


Figure 3.5 Continued

Subroutine ACHFUN

Subroutine ACHFUN is the module which contains the functions of weighted positive and negative deviation variables. These functions must be supplied by the user for each new NLGP problem. An example will be presented in the next chapter which will show these functions, one for each priority level in the achievement vector. Figure 3.6 shows the computer flow diagram for this subroutine.

Variables and parameters used in ACHFUN:

A
N
P

Subroutine DEVVAR

Subroutine DEVVAR calculates the values of the negative and positive deviation variables. It uses the values of the objective functions already calculated and compares these values to the desired aspiration levels, RHS. Figure 3.7 shows the flow diagram.

Variables and parameters used in DEVVAR:

NOBJ
OBJ
RHS
N
P

Subroutine OBJFUN

Subroutine OBJFUN is used to calculate the values of the objective functions at the test point x. These functions must be user supplied for each new NLGP problem to be solved. An example of this subroutine will be presented in the next chapter. A flow diagram of this subroutine is contained in Figure 3.8.

Variables and parameters used in OBJFUN:

OBJ
X

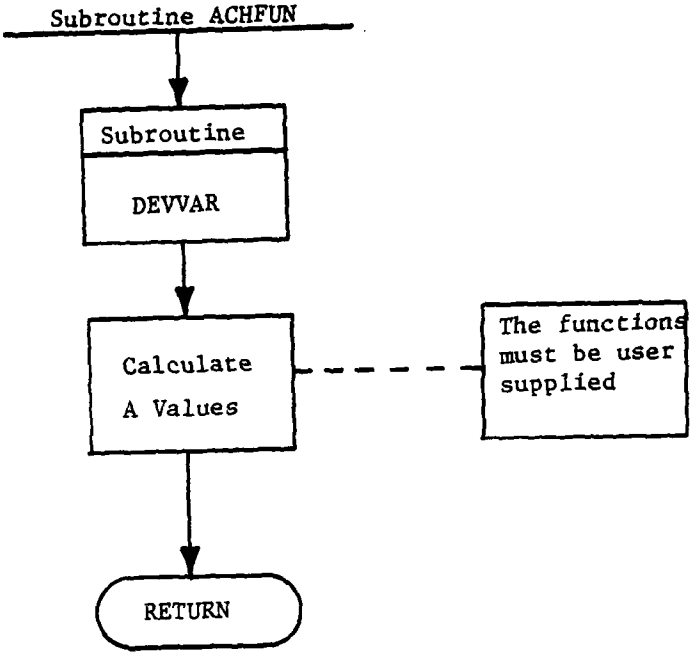


Figure 3.6 Flow Diagram for Subroutine ACHFUN

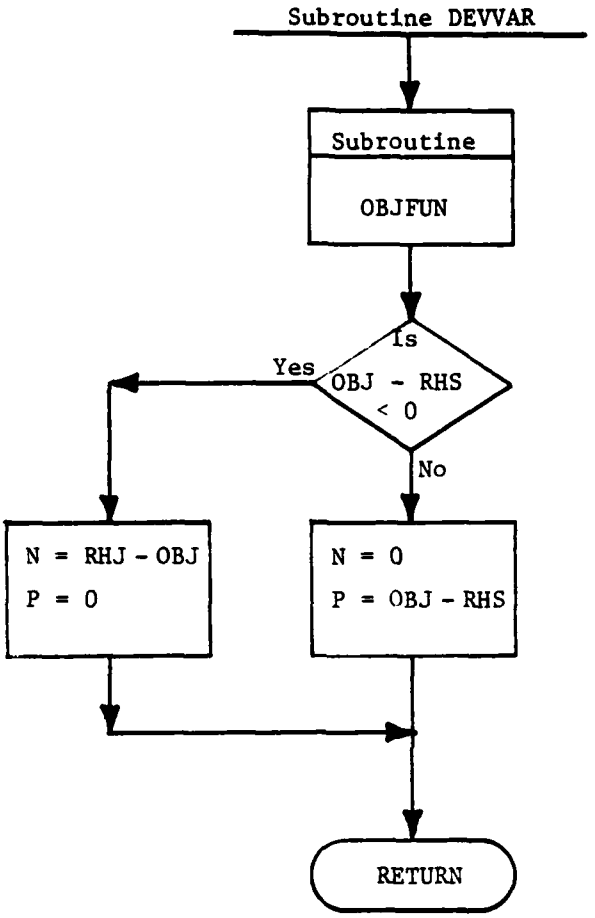


Figure 3.7 Flow Diagram for Subroutine DEVVAR

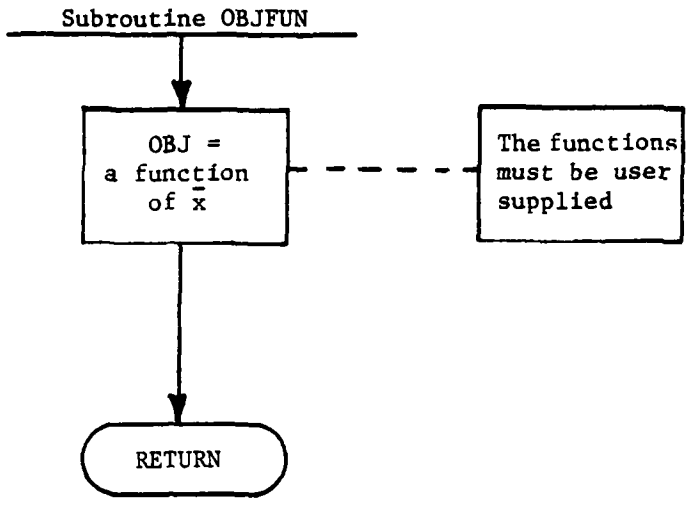


Figure 3.8 Flow Diagram for Subroutine OBJFUN

3.3 Modifications

The second and third versions of the NLGP/MPS-RS code are modifications of version 1. The modifications which make them different from the basic code are presented here. Otherwise, these codes are exactly the same as version 1. Complete FORTRAN language computer listings of these versions are contained in Appendices B and C.

The NLGP/MPS-RS: Version 2 Code

This computer code is exactly like version 1 except that the perturbation step sizes, EPS and IPS, are vectors instead of scalars. This enables the user to input a separate step size for each variable. This can be especially important when the range of values which the variables can take on vary considerably.

The major difference in the code from version 1 is in subroutine DATAIN. Now a vector of values for EPS must be read into storage. Table 3.4 lists the necessary cards for input for version 2.

Minor modifications also appear in subroutines HJALG and RIDGE. All statements previously using IPS and EPS now use the variable forms, IPS(k) and EPS(k). The resetting of the step size vector in subroutine RIDGE is accomplished using a do loop of the form:

```
DO 10 K=1, NVAR
IPS(k) = EPS(k).
```

Advantages for using this version as compared to the other versions will be presented in the next section.

The NLGP/MPS-RS: Version 3 Code

Version 3 of the computer code is the same as version 2 with an additional subroutine, subroutine UVALUE. This subroutine has been added to reduce the computation time necessary for the evaluation of the objective functions.

Table 3.4 Data Input Guide NLGP/MPS-RS Code: Version 2

Card(s)	Data	Type	Format
1	NOBJ	Integer *4	I5
	NPRIOR	"	I5
	NVAR	"	I5
	NMAX	"	I5
	IPRINT	Integer *2	I5
2 - $(\frac{NVAR}{8} - 1)$	X	Real *4	8F10.0
3 - $(\frac{NVAR}{8} - 1)$	EPS	Real *4	8F10.0
4 - $(\frac{NVAR}{8} - 1)$	LBD	Real *4	8F10.0
5 - $(\frac{NVAR}{8} - 1)$	UBD	Real *4	8F10.0
6 - $(\frac{NOBJ}{8} - 1)$	RHS	Real *4	8F10.0
7 - $(\frac{NPRIOR}{8} 1)$	EPSY	Real *4	8F10.0
8	ACCEL	Real *4	F10.0
	REDUCE	"	F10.0
	DELTA	"	F10.0

Whenever a new test point, X, is formed, subroutine OBJFUN is eventually called to evaluate all the objectives at this test point so that a new achievement vector can be formed. There are many cases, however, when it is not necessary to re-evaluate all the objective functions. There are two reasons for this:

(1) Many problems have objectives which are sparsely populated by the variables. It is not necessary to re-calculate all the objectives when the variable being perturbed only appears in a few of the objectives, and

(2) Since the pattern search algorithm perturbs one variable at a time during exploratory moves, it is not necessary to re-calculate every term in each objective function. Rather, only those terms in which the perturbed variable appears need to be recalculated, and the net effect on the value of the objective function can be determined.

Both categories occur frequently in large-scale nonlinear problems. The use of the UVALUE subroutine eliminates many unnecessary calculations. This results in significant reductions in computation time, especially in large problems. The subroutine can be especially useful during the exploratory moves.

Additionally, this subroutine is problem dependent and must be supplied in part by the user for implementation with the code. A flow chart description of subroutine UVALUE is given here for the target allocation problem described in Chapter II. A complete description of the use and set-up of this subroutine is contained in the User's Manual.

The major modifications contained in this code will now be presented. The new parameters and variables which are contained in version 3 are listed in Table 3.5. There use in the program will be indicated through flow charts showing the revised subroutines.

Subroutine DATAIN

Subroutine DATAIN is exactly the same as that used in version 1 with one addition. The matrix of coefficients, AMAP(I,L) must be read into storage. It is currently dimensioned to (100,100). Larger problems could be solved by increasing this matrix in subroutine UVALUE and DATAIN. Table 3.6 shows the revised data input guide for version 3.

Subroutine HJALG

Subroutine HJALG in version 3 differs from that in version 1 in that it now calls subroutine UVALUE instead of subroutine OBJFUN in many places in the program, as indicated by the switch, JPI. Subroutine OBJFUN is called initially to calculate the values of the objective functions. During the exploratory moves, subroutine UVALUE is used to calculate the incremental changes in the objective functions for the variable being perturbed. The additional variables and parameters used in subroutine HJALG are:

JPI
LNUM
UVALUE
SEPS.

The complete flow diagram for this subroutine is in Figure 3.9.

Subroutine RIDGE

Same as in version 1.

Subroutine DECIDE

Same as in version 1.

Table 3.5 Additional Parameters and Variables for NLGP/MPS-RS Code: Version 3

Name	Description
LNVM	Counter used to indicate which variable K is being perturbed in the exploratory moves
DIFF	A vector of the incremental changes in the objective functions
JPI	A switch indicator: if JPI = 0 subroutine UVALUE is not used, if JPI = 1, subroutine UVALUE is called to calculate DIFF for the objectives
AMAP	A matrix of coefficients for the objective functions
SEPS	An adjusted step size scalar
A,B,C,D,E	Subroutine UVALUE internal variables: problem dependent on actual use

Table 3.6 Data Input Guide NLGP/MPS-RS
Code: Version 3

Card(s)	Data	Type	Format
1	NOBJ	Integer #4	I5
	NPRIOR	"	I5
	NVAR	"	I5
	NMAX	"	I5
	IPRINT	"	I5
	IU	"	I5
2 - $(\frac{NVAR}{8} - 1)$	X	Real #4	8F10.0
3 - $(\frac{NVAR}{8} - 1)$	LBD	Real *4	8F10.0
4 - $(\frac{NVAR}{8} - 1)$	UBD	Real *4	8F10.0
5 - $(\frac{NOBJ}{8} - 1)$	RHS	Real *4	8F10.0
6 - $(\frac{NPRIOR}{8} - 1)$	EPSY	Real *4	8F10.0
7 -	EPS	Real *4	F10.0
	ACCEL	"	F10.0
	REDUCE	"	F10.0
	DELTA	"	F10.0
8 - $(\frac{NVAR \times NOBJ}{8} - 1)$	AMAP	Real *4	8F10.0

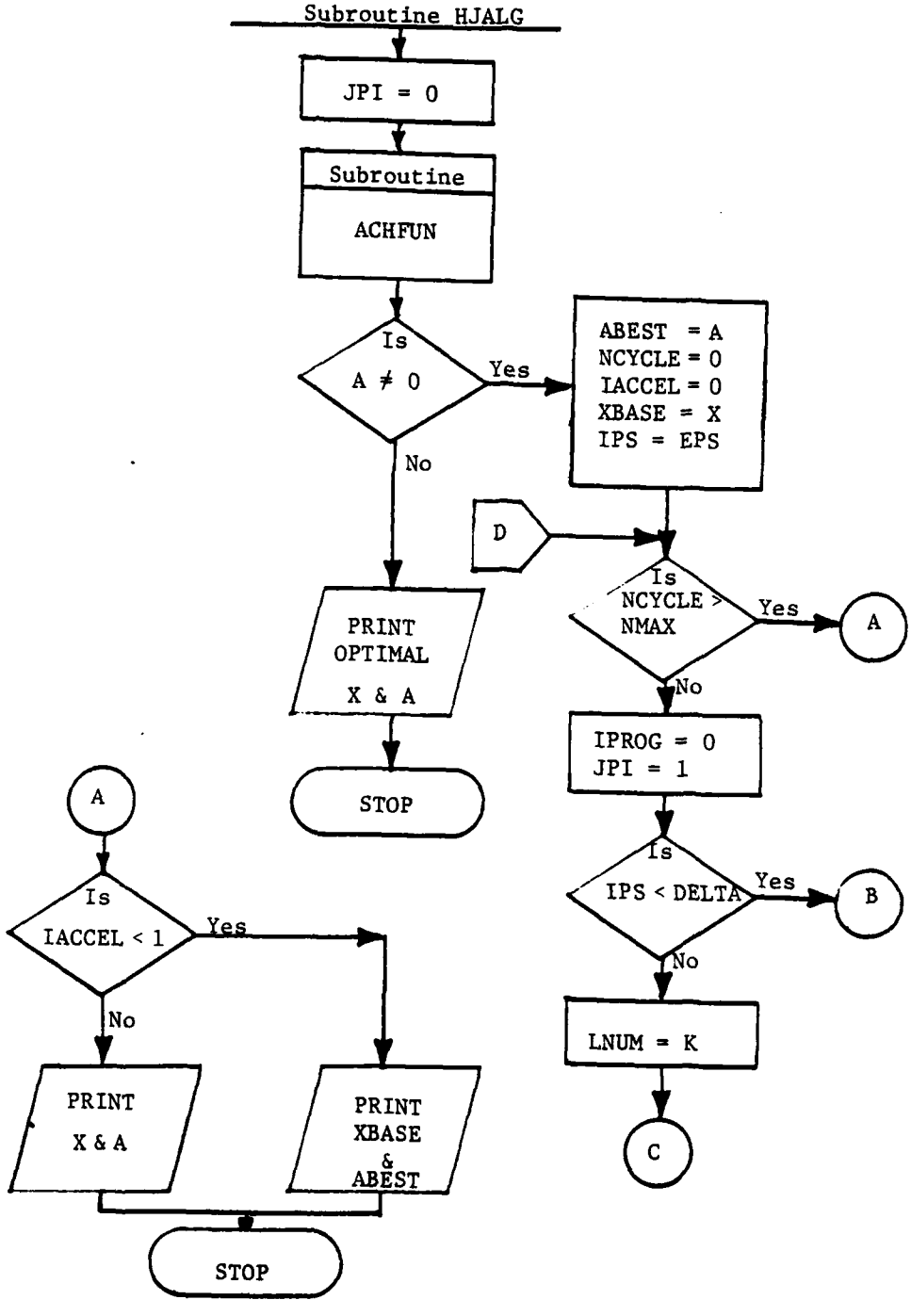


Figure 3.9 Flow Diagram for Subroutine HJALG: Version 3

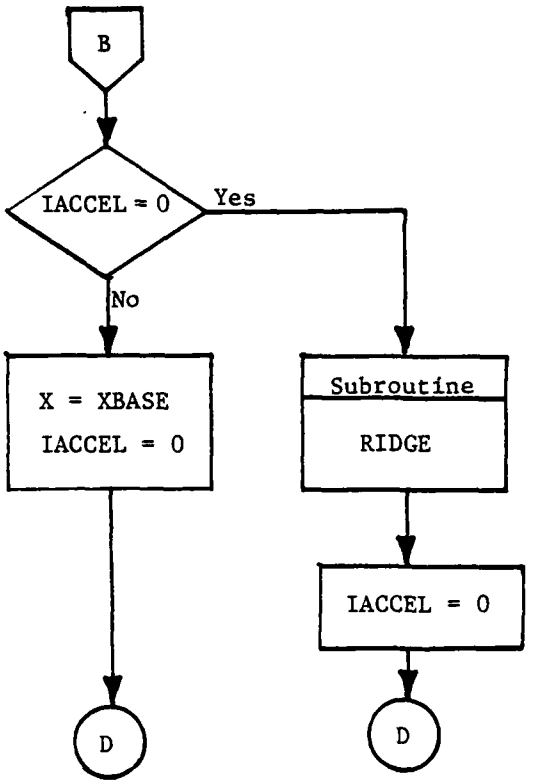


Figure 3.9 Continued

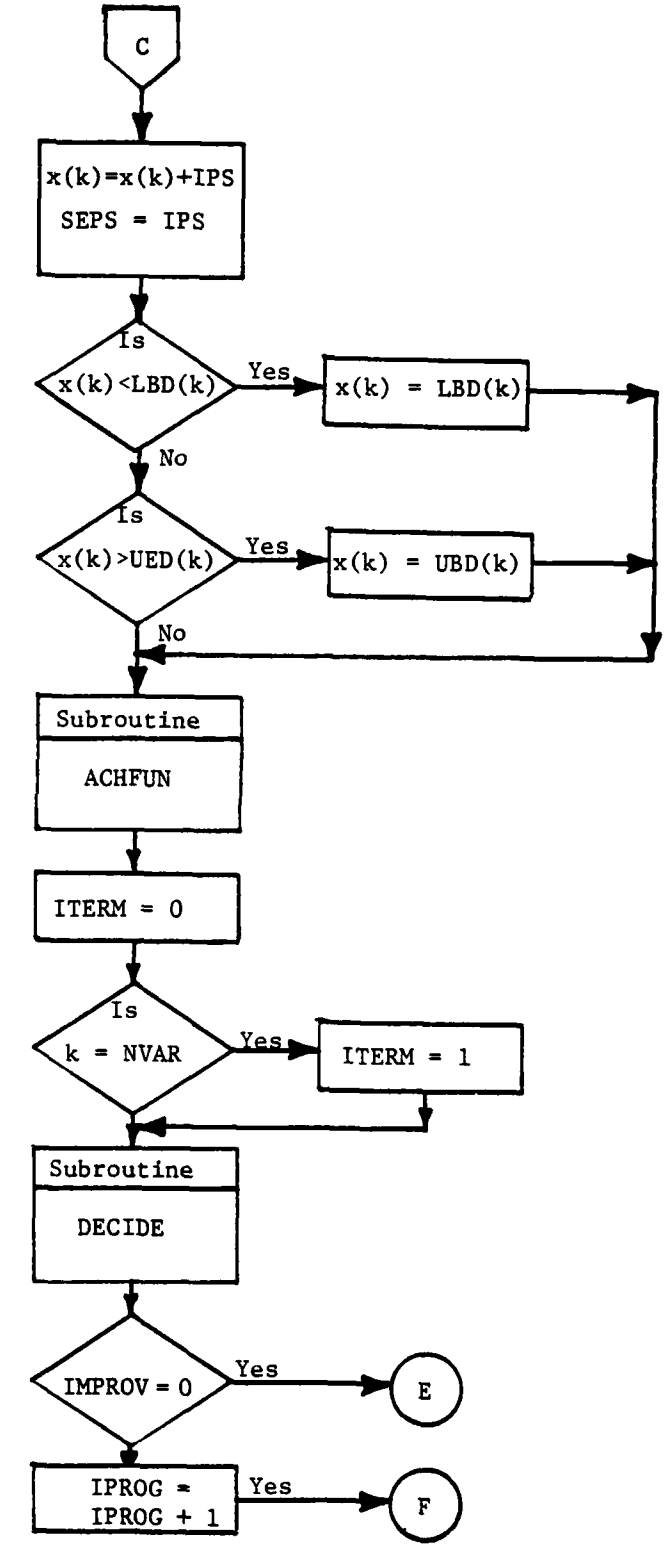


Figure 3.9 Continued

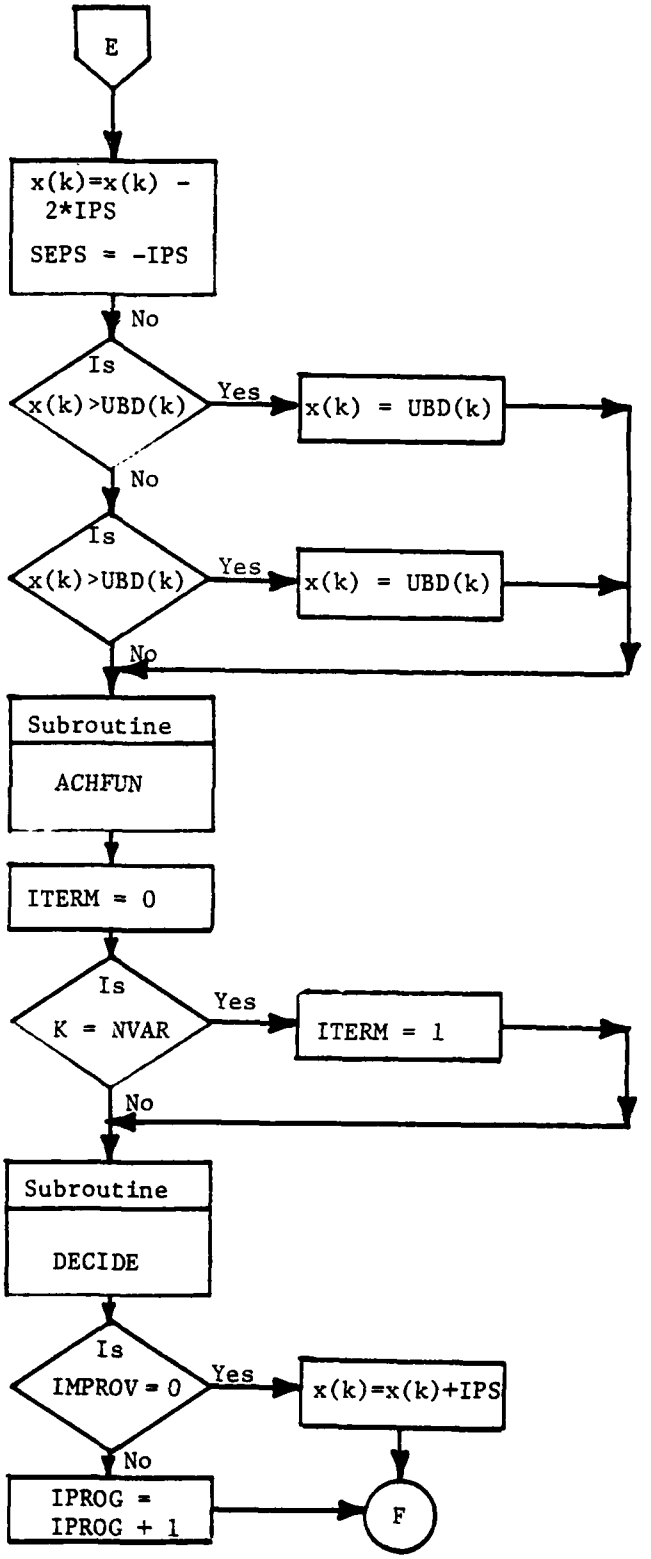


Figure 3.9 Continued

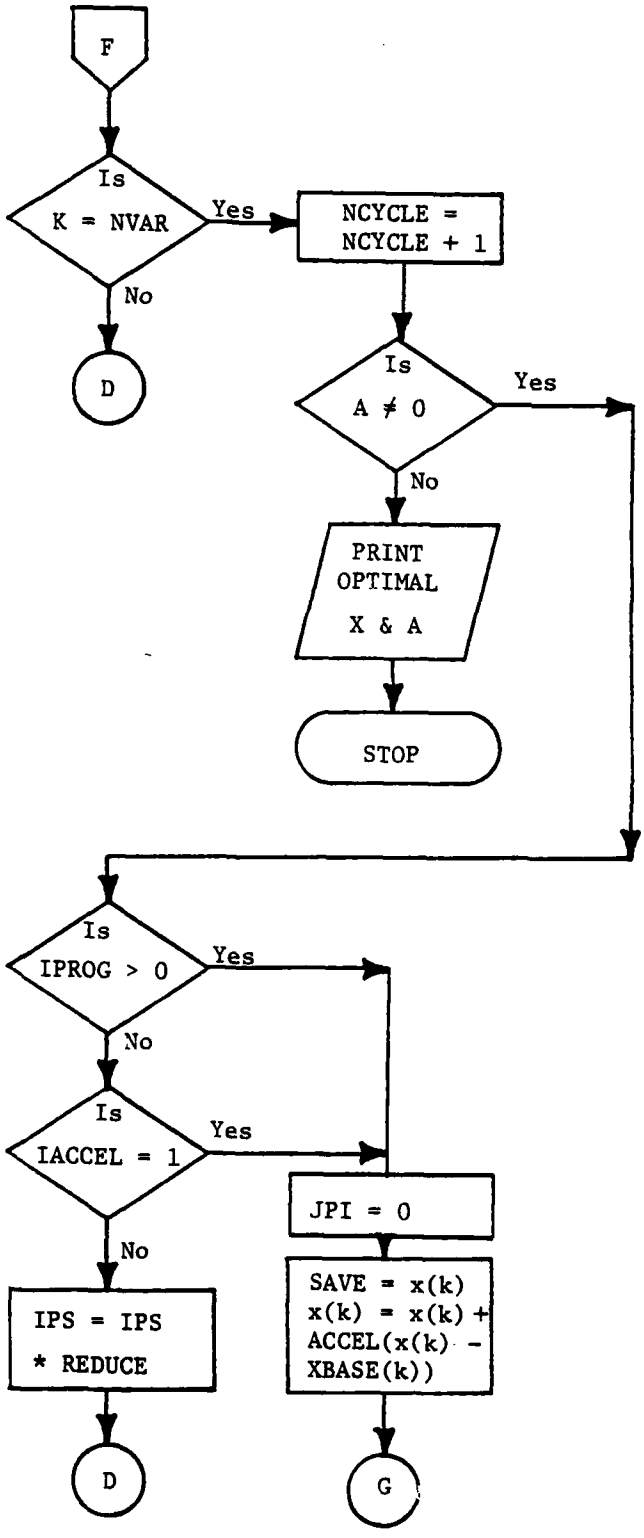


Figure 3.9 Continued

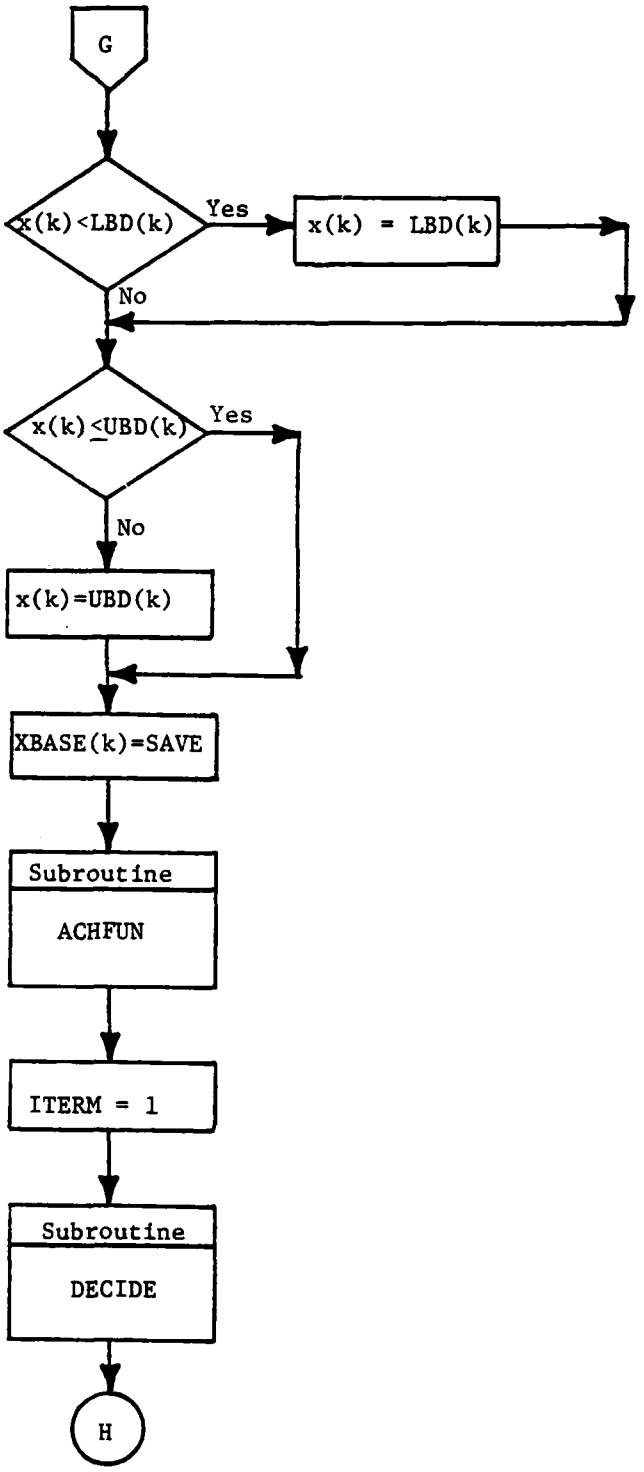


Figure 3.9 Continued

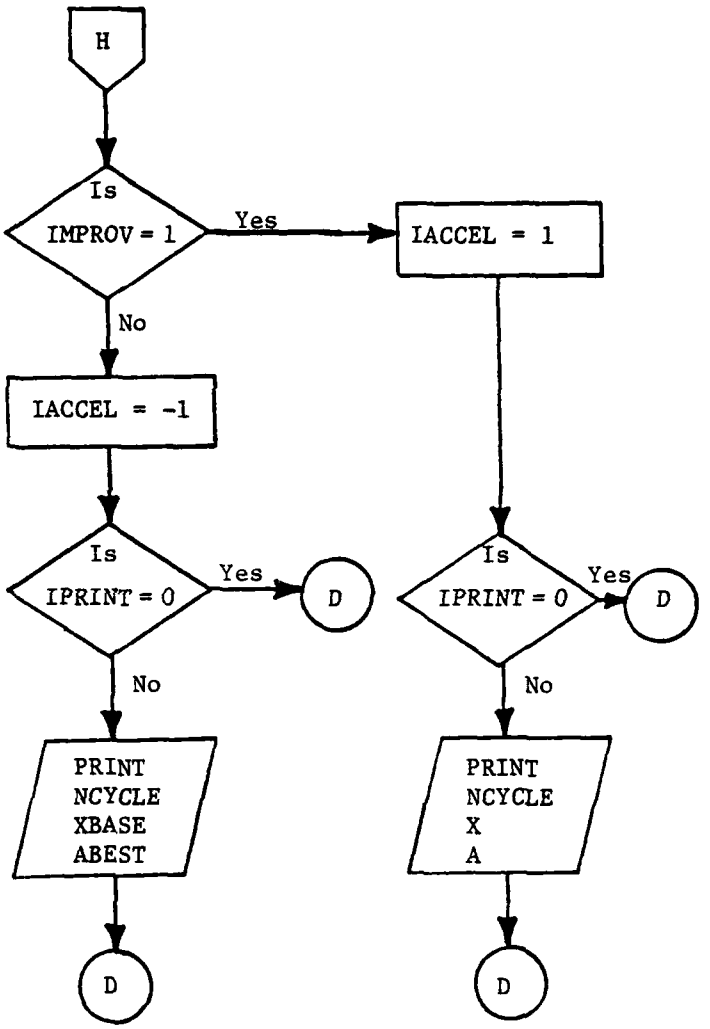


Figure 3.9 Continued

Subroutine ACHFUN

Same as in version 1 (problem dependent).

Subroutine DEVVAR

This subroutine is slightly different, in that it now tests to see if subroutine OBJFUN is UVALUE is to be called, then calls the appropriate one. Figure 3.10 shows the flow diagram for this subroutine.

Subroutine OBJFUN

Same as in version 1 (problem dependent).

Subroutine UVALUE

This subroutine calculates the incremental changes in the objective functions. The variables and parameters used in this subroutine are:

NOBJ	SEPS
L	A,B,C,D,E
LNUM	Y9
DIFF	
AMAP	

Calls to other subroutines: None.

The flow diagram for this subroutine is contained in Figure 3.11. It should be restated that this subroutine has many statements which are problem dependent, and those shown in the flow diagram represent a schematic for the objectives used in the target allocation problem of Chapter II. A more thorough discussion of the formulation of this subroutine is given in the User's Manual.

Further discussion of the three versions and their respective advantages and limitations will be given in the following sections of Chapter III.

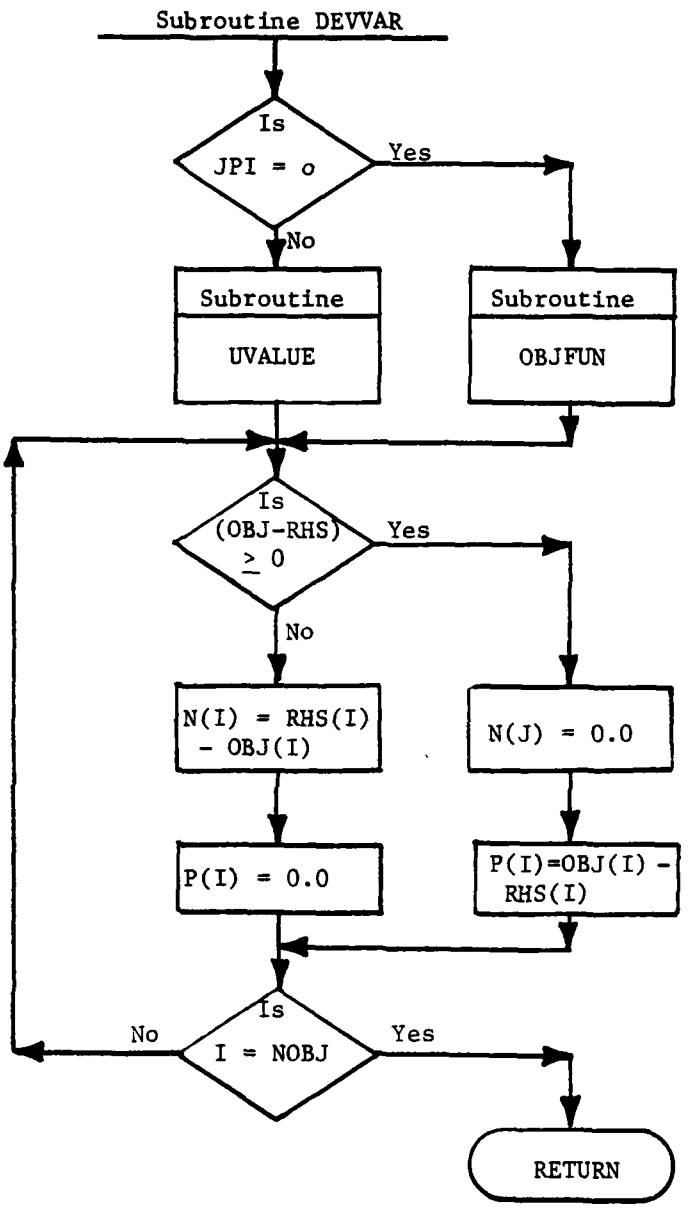


Figure 3.10 Flow Diagram for Subroutine DEVVAR: Version 3

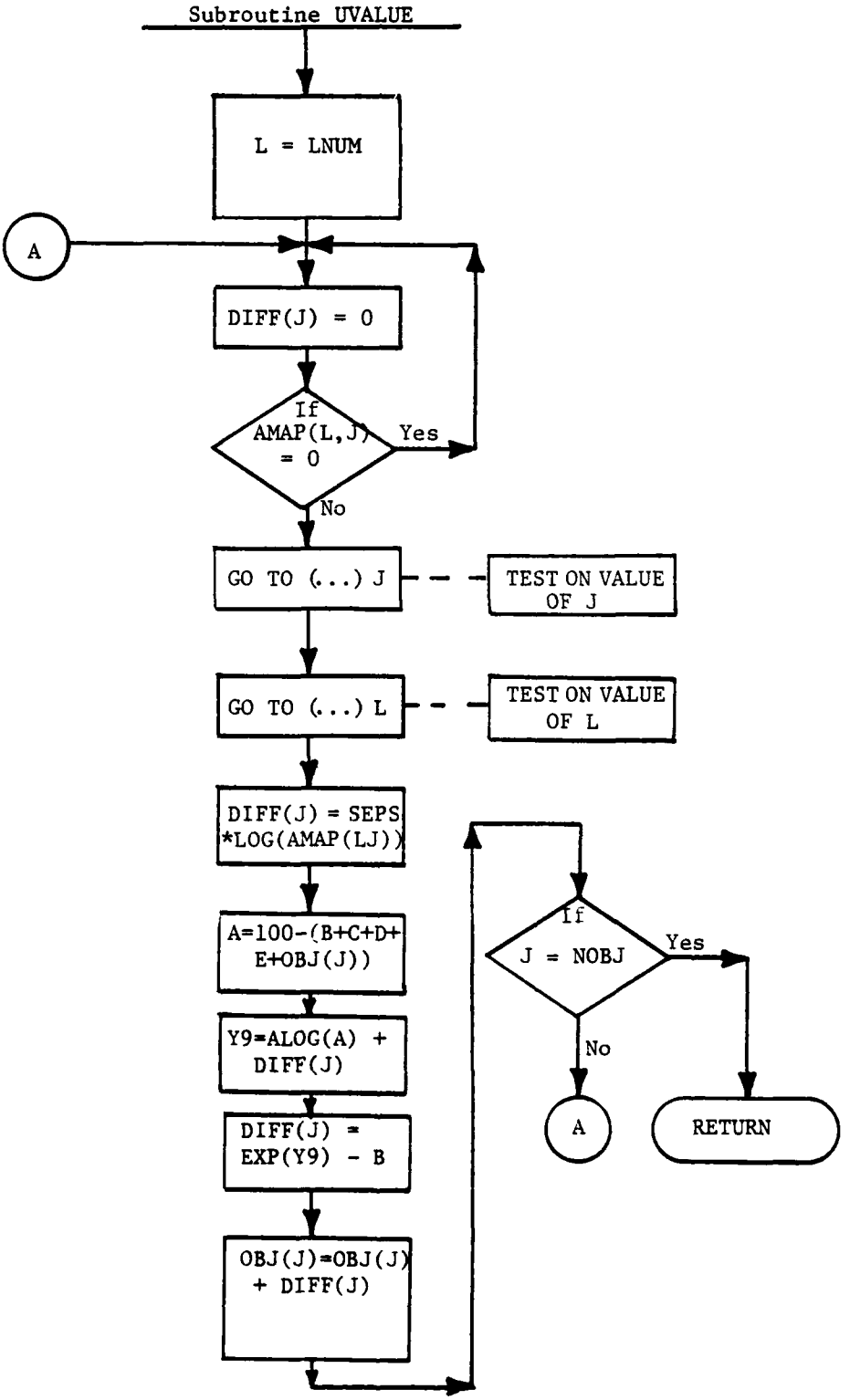


Figure 3.11 Flow Diagram for Subroutine UVALUE

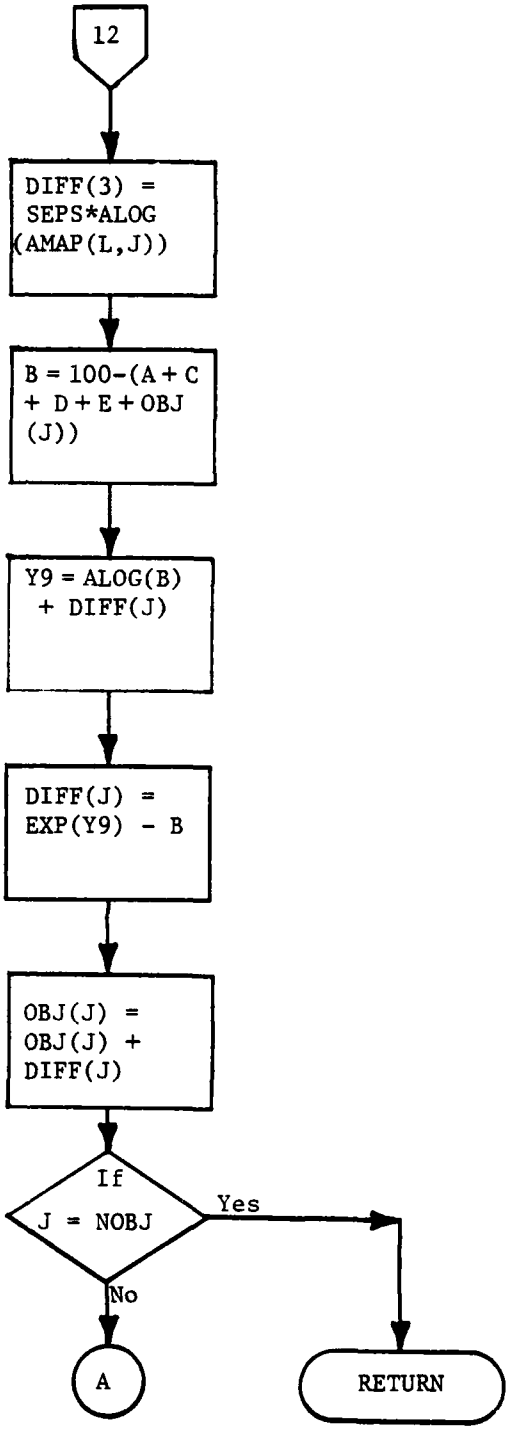


Figure 3.11 Continued

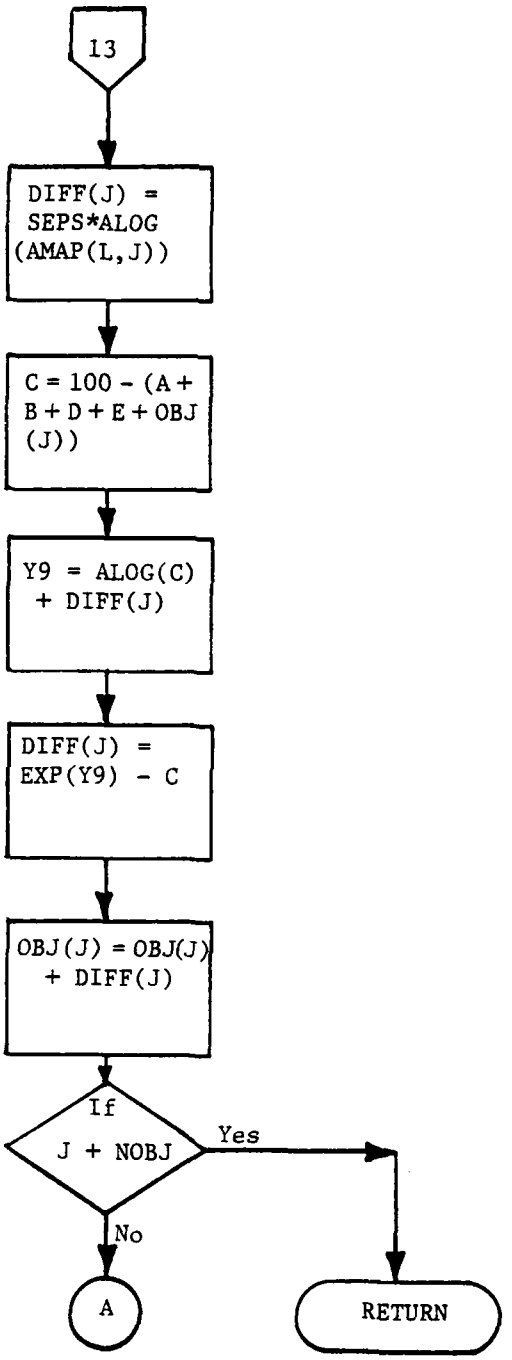


Figure 3.11 Continued

3.4 Specifications

Each version of the code has different computer storage specifications, due to the modifications contained in versions 2 and 3. All three codes are presently dimensioned for problems of the following maximum size:

Number of objectives = 2500
Number of variables = 2500
Number of priority levels = 10 .

The computer core storage necessary for each version of the code will be dependent upon the number and type of objective functions and the number of and term in the achievement vector. If every variable appears in every objective function, the core storage necessary for subroutine OBJFUN could be conservatively estimated to be:

(80 Bytes/card) x (1 card/10 variables) x (2500 variables)

= 20,000 Bytes/objective

(20,000 Bytes/objective) x (2500 objectives)

= 50 million Bytes.

This is an estimated minimum storage requirement <u>if</u> <u>all</u> variables appear in all objectives. Since most problems do not have this form, actual storage will be much smaller. Table 3.7 gives the requirements which are minimum values, and actual requirements will vary greatly depending on the problem.

3.5 Advantages and Limitations

There are advantages to using each code which depend on the type of problem to be solved. Version 1 is the basic code, the simplest in form, and the easiest to use. This code is differentiated from the other two versions in that it uses a scalar for the step size, EPS, for the perturbations in the exploratory moves. This is advantageous when the decision

Table 3.7 Storage Requirements

	Bytes (in decimal)	
Version	Object Code	Array Area
1	22,016	80,260
2	22,016	90,260
3	22,784	130,308

variables for the problem are all similar in nature and in the range of values which they can have. Using the scalar step size saves 10,000 bytes of core storage. This area can be used to redimension the number of variables to sizes larger than 2500.

Version 1 is limited in this same respect if the range of values for the variables vary greatly or if their very nature makes it difficult to select only a single step size for the exploratory moves. This disadvantage is eliminated by the use of version 2. With a vector of step sizes, each variable can have a distinct step size to account for the type and range of values peculiar to it. This results in an additional cost of 10,000 bytes of core storage, but only a negligable increase in computation time.

The third version of the code is the most complex, and therefore, the most difficult for the user to implement. All codes must have user supplied subroutines OBJFUN and ACHFUN, but version 3 also requires a user supplied subroutine UVALUE. This is the most difficult subroutine to formulate. Its advantages of saving considerable computer time offsets the disadvantages of its additional storage requirements and difficulty of use only when the problem to be solved is large (i.e., in the range of 1000 or more variables and 500 or more objectives) and when only a few variables appear in each objective. It is the opinion of the authors that the use of version 3 be attempted only after the user has gained experience using versions 1 and 2, and also has a thorough understanding of the problem to be solved.

The direct search technique utilized in each version is superior to gradient search techniques in the number and type of evaluations which must be made to calculate an achievement vector. No gradients have to be calculated or estimated through expansion of series.

This technique is limited in that one can only hope to obtain a local optimal for the problem solution, especially for large and complex problems. (Global optimal solutions can be obtained, however, when the problem is of a special nature and limited complexity.)

Finally, the computer codes are designed to find solutions which are continuous - valued in nature. Problems which require solutions with 0-1 valued variables and mixed integer-continuous problems present special difficulties. No guarantee can be made that the solutions obtained will be integer valued. Some attempt can be made to control the variable values by restricting the lower and upper bounds and restricting the step size used for variable perturbations.

More comments on these considerations will be made in the final chapter.

3.6 Summary

Three different versions of the NLGP/MPS-RS computer code have been developed. Each version has advantages over the others which makes it superior for solving a particular type or class of NLGP problems.

A brief description has been given for each code, explaining its general form and the types of variables and parameters used in each. Flow diagrams depict the subroutines composing each code, and an overall flow schematic shows the relationships of these subroutines to form an entire computer program. Listings for each are given in Appendices A, B, and C.

Some general advantages and limitations for the use of each version have been described. More specific information detailing the implementation of each code can be found in the User's Manual supplied along with this report. Considerations on problem solution and computational aspects will be given in Chapter IV.

CHAPTER IV

COMPUTATIONAL RESULTS

4.1 Introduction

The general nonlinear goal programming problem formulation was presented in Chapter II along with an example. The three variations of the computational techniques used to solve such an NLGP problem were described in Chapter III. The solution to the example problem will be presented here first, and then the remainder of this chapter will discuss computational results in general.

4.2 A Target Allocation Problem: Solution

The target allocation problem formulated in Chapter II is a small-scale problem, but it shows a type of problem suitable to naval application. The NLGP model for this problem is:

Find the values of $\bar{x} = (x_1, x_2, \ldots, x_{24})$ so as to minimize $\bar{a} = (p_1 + p_2 + p_3 + p_4 + n_5 + n_6 + n_7 + n_8, n_9)$ subject to:

Aircraft availability objectives

$$\text{Goal 1:} \quad x_1 + x_3 + n_1 - p_1 = 27$$

$$\text{Goal 2:} \quad x_2 + x_4 + n_2 - p_2 = 102$$

Carrier space objectives

$$\text{Goal 3:} \quad 2920\, x_1 + 1770\, x_2 + n_3 - p_3 = 112{,}300$$

$$\text{Goal 4:} \quad 2920\, x_3 + 1700\, x_4 + n_4 - p_4 = 147{,}100$$

Limitations on missions due to aircraft availability

$$\text{Goal 5:} \quad x_1 - 0.6452\,(x_5 + x_7 + x_9) - 0.06250\,(x_6 + x_8) + n_5 - p_5 = 0$$

$$\text{Goal 6:} \quad x_2 - 0.05556\,(x_{10} + x_{12} + x_{14}) - 0.05264\,(x_{11} + x_{13}) + n_6 - p_6 = 0$$

$$\text{Goal 7:} \quad x_3 - 0.06896\,x_{15} - 0.06452\,(x_{16} + x_{19}) - 0.06250\,(x_{17} + x_{19}) + n_7 - p_7 = 0$$

$$\text{Goal 8:} \quad x_4 - 0.05882\,x_{20} - 0.0556\,(x_{21} + x_{24}) - 0.05264\,(x_{22} + x_{23}) + n_8 - p_8 = 0$$

Damage objective

$$\text{Goal 9:} \quad 40\,(1 - 0.99978^{(x_5 + x_{15})}\,0.99953^{(x_{10} + x_{20})}) + 10\,(1 - 0.99978^{(x_6 + x_{16})}\,0.99953^{(x_{11} + x_{21})}) + 50\,(1 - 0.99978^{(x_7 + x_{17})}\,0.99953^{(x_{12} + x_{22})})$$

and $\quad \bar{x}, \bar{n}, \bar{p} \geq \bar{0}$

This problem has the following specifications for the computer codes:

```
NOBJ   =   9
NVAR   =  24
NPRIOR =   2.
```

The objectives are divided into two groups for the priority levels. Priority level one contains the first eight objectives. These are resource objectives which cannot be violated. The second priority level measures the damage inflicted on all targets by the aircraft from the two aircraft carriers. The reader is referred to Chapter II for a complete description of all problem parameters.

Table 4.1, 4.2, and 4.3 list all the input information necessary for the solution of this problem. Table 4.1 contains the program parameters necessary for control of the pattern search. Table 4.2 contains data related to the variables, and Table 4.3 contains the data as related to the objectives and achievement vector.

The solution to this problem is given in Table 4.4 The achievement vector element a(1) = 0.0 indicates that the first eight objectives (i.e., constraints) have all been satisfied. The second element a(2) measures the total damage to enemy targets in terms of the military value placed on each target set. The maximum attainable damage would be 100 units. The value a(2) = 67.55 is the underachievement from this maximum aspiration level. Thus, the difference (100 - 67.55) = 32.45 units is the measure of total damage inflicted on all enemy targets.

The decision variables represent the best solutions for $\bar{x}$ to attain the values of a(1) and a(2). The reader is referred to Table 4.5 for the conversion of the solution to the actual problem variables.

The solution which has been calculated has continuous values for the decision variables. The nature of the problem and the variables demands that they be integer values. All non-integer values appear to be the result of round-off errors in the calculations. The solution at integer values rounded to the next closest value is:

$$a(1) = 0.0$$
$$a(2) = 67.71$$

This results in a military value for the damage of 32.29, which is slightly less than the continuous solution and differs by 0.49 percent. The computer code gives an acceptable, if not exact, solution to the NLGP problem.

Table 4.1 Input Data for Program Parameters: Target Allocation Problem

Number of Objectives =	9
Number of Priorities =	2
Number of Decision Variables =	24
Maximum Number of Allowable Pattern Search Cycles =	50
Initial Step Sizes =	2.0000
Acceleration Factor =	2.0000
Step Size Reduction Factor =	0.5000
Minimum Allowable Step Size =	0.0001

Table 4.2 Input Information for Decision Variables: Target Allocation Problem

VECTOR OF INITIAL ESTIMATES FOR DECISION VARIABLES---X		Lower and Upper Bounds on Decision Variables---LBD and UBD			
X(1) =	22.00000	LBD(1) =	0.0	UBD(1) =	30.00000
X(2) =	37.00000	LBD(2) =	20.00000	UBD(2) =	60.00000
X(3) =	15.00000	LBD(3) =	0.0	UBD(3) =	30.00000
X(4) =	75.00000	LBD(4) =	40.00000	UBD(4) =	80.00000
X(5) =	150.00000	LBD(5) =	30.00000	UBD(5) =	300.00000
X(6) =	2.00000	IBD(6) =	0.0	UBD(6) =	5.00000
X(7) =	10.00000	LBD(7) =	10.00000	UBD(7) =	30.00000
X(8) =	0.0	LBD(8) =	0.0	UBD(8) =	400.00000
X(9) =	0.0	LBD(9) =	0.0	UBD(9) =	10.00000
X(10) =	600.00000	LBD(10) =	500.00000	UBD(10) =	1000.00000
X(11) =	2.00000	LBD(11) =	0.0	UBD(11) =	5.00000
X(12) =	10.00000	LBD(12) =	0.0	UBD(12) =	30.00000
X(13) =	0.0	LBD(13) =	0.0	UBD(13) =	500.00000
X(14) =	0.0	LBD(14) =	0.0	UBD(14) =	10.00000
X(15) =	40.00000	LBD(15) =	20.00000	UBD(15) =	70.00000
X(16) =	2.00000	LBD(16) =	0.0	UBD(16) =	5.00000
X(17) =	100.00000	LBD(17) =	50.00000	UBD(17) =	300.00000
X(18) =	0.0	LBD(18) =	0.0	UBD(18) =	100.00000
X(19) =	0.0	LBD(19) =	0.0	UBD(19) =	10.00000
X(20) =	10.00000	LBD(20) =	0.0	UBD(20) =	30.00000
X(21) =	2.00000	LBD(21) =	0.0	UBD(21) =	5.00000
X(22) =	1200.00000	LBD(22) =	800.00000	UBD(22) =	1500.00000
X(23) =	0.0	LBD(23) =	0.0	UBD(23) =	100.00000
X(24) =	0.0	LBD(24) =	0.0	UBD(24) =	10.00000

Table 4.3 Input Data for Objectives and Achievement Vector: Target Allocation Problem

Right-hand side values for objective functions---RHS

```
RHS (1)  =          27.0000
RHS (2)  =         102.0000
RHS (3)  =      112300.000
RHS (4)  =      147100.000
RHS (5)  =           0.0
RHS (6)  =           0.0
RHS (7)  =           0.0
RHS (8)  =           0.0
RHS (9)  =         100.0000
```

Vector of achievement function tolerances---EPSY

```
EPSY (1)  =  0.0000100
EPSY (2)  =  0.0000100
```

Table 4.4 Target Allocation Problem Solution

Decision Variables

X (1)	=	14.999939
X (2)	=	33.999939
X (3)	=	9.999939
X (4)	=	63.999939
X (5)	=	126.999939
X (6)	=	0.999939
X (7)	=	10.999939
X (8)	=	6.999939
X (9)	=	6.999939
X (10)	=	576.000000
X (11)	=	0.999939
X (12)	=	0.999939
X (13)	=	0.999939
X (14)	=	0.999939
X (15)	=	20.000000
X (16)	=	0.999939
X (17)	=	76.999939
X (18)	=	20.999939
X (19)	=	8.999939
X (20)	=	0.999939
X (21)	=	0.999939
X (22)	=	1176.00000
X (23)	=	24.999939
X (24)	=	8.999939

Achievement Values

A (1)	=	0.0
A (2)	=	67.554916

Table 4.5 Target Allocation Problem Solution

Description		Problem Variables	Solution for Variable
Distribution of aircraft of type j to carrier i		x_{11}	14.999
		x_{12}	33.999
		x_{21}	39.999
		x_{22}	63.999
Missions flown from carrier 1 to target k	Using aircraft of type 1	y_{111}	126.999
		y_{112}	0.999
		y_{113}	10.999
		y_{114}	6.999
		y_{115}	6.999
	Using aircraft of type 2	y_{121}	576.000
		y_{122}	0.999
		y_{123}	0.999
		y_{124}	0.999
		y_{125}	0.999
Missions flown from carrier 2 to target k	Using aircraft of type 1	y_{211}	20.000
		y_{212}	0.999
		y_{213}	76.999
		y_{214}	20.999
		y_{215}	8.999
	Using aircraft of type 2	y_{221}	0.999
		y_{222}	0.999
		y_{223}	1176.000
		y_{224}	24.999
		y_{225}	8.999

Anytime a problem is encountered which must have an integer solution for some of the variables, the non-integer values can be rounded to integer values and resolved for a problem solution. This method cannot guarantee that a better integer solution does not exist. This method should result in a solution which is close in value to the best continuous solution to the problem. It also yields a good starting point for the examination of other integer solutions.

For comparison, Rice, et al [28] obtained similar results for the total damage inflicted using SUMT [5]. Using the target values for scenario 1, their solution had 33.67 percent of the total target value damaged, as compared to 32.45 percent using the NLGP method. They obtained assignments of 9 aircraft of type 1 and 69 aircraft of type 2 to carrier 2. Unfortunately, this solution violates the carrier space availability constraint for carrier 2 (148,410 ft^2 used versus 147,100 ft^2 available). This makes their solution infeasible. The NLGP solution, although slightly smaller in value for the damage percentage, is better since it does yield a feasible solution. No priority level one objectives (i.e., resouce constraints) were violated. The authors feel that in more complex and large problems, the superiority of the NLGP method over single objective methods will be more pronounced.

4.3 Empirical Results

The relationship of problem size (i.e., the number of objectives and number of variables) versus computation time and storage requirements has been examined. The results of many problems which were solved will be presented on a aggregate level.

Testing of the three computer codes proceeded from an initial phase where problems were small and simple in nature to a secondary phase to the final phase where problem size exceeded 100 variables and 100 objectives.

In the initial phase, a number of problems with less than 10 objectives and 10 variables were solved. The second phase of testing involved problems ranging in size from 10 to 25 objectives and 25 to 75 variables. The last phase involved problems with over 100 objectives and over 100 variables. A summary of the results is in Table 4.6.

The average compilation time increases because the object code increases with problem size and complexity. This is because the user supplies subroutines OBJFUN and ACHFUN (and UVALUE, if version 3 is used). These will vary in size by problem.

The average execution time of course increases with increases in problem size. It is interesting that the difference between the last two categories is slight. This is due in part to the fact that the problems with greater than 100 objectives and 100 variables had primarily linear objectives (usually 100) and only a few nonlinear objectives (10 to 20). This made execution of these problems much faster than would normally be expected with the majority being nonlinear objectives.

Computer storage versus problem size is a difficult relationship to predict. The array area is fairly constant as would be expected, since this area depends on the dimensioning within the program. The object code sizes varied and tended to increase with increases in problem size. This storage requirement will be highly dependent on the type of objectives (i.e., linear versus nonlinear), complexity of the objectives (i.e., quadratic terms, exponential terms, logarithmic terms), the number of terms in each objective, and the number of objectives in the problem. These areas will dictate storage requirements, and also, execution time.

Table 4.6 Computation Results

Problem Size	TIME (SECONDS)			STORAGE (BYTES)	
	Average Compile Time	Average Execution Time	Range Execution Time	Object Code (AVE)	Array Area (AVE)
NOBJ≤10 NVAR≤10	0.230 secs.	0.064 secs.	0.04 - 0.11 secs.	17476 Bytes	80260 Bytes
NOBJ≤25 25≤NVAR≤50	0.248	1.217	0.64 - 2.11	22746	80260
NOBJ≤25 50≤NVAR≤75	0.349	9.591	1.72 - 17.32	18062	86473
100≤NOBJ≤150 100≤NVAR≤150	0.930	10.260	9.56 - 11.44	47931	80260

Note: All jobs were run on an IBM 370/3033 system using the FORTRAN WATFIV compiler.

4.4 Summary

All three codes have been tested and give the same solutions to the same problems. Execution times of the programs are fairly equivalent for all three codes. Version 3 had no particular advantage over the others for the problems on which it was tested, but those problems were deemed too small to show appreciable advantages. Version 3 should give substantial savings in the area of execution time over versions 1 and 2 when the problem size is large and the objectives are sparsely populated by the variables.

Version 1 is the least complicated of the three codes and easiest to use. Version 2 is the next in line in terms of ease of access. Version 3 is the most difficult to set up for computation due to subroutine UVALUE. Its advantages have already been mentioned.

CHAPTER V

CONCLUSIONS

5.1 Conclusions

The specific objectives of this study have been met. To meet the demands for a general computer tool to solve a broad range of NLGP problems, three computer codes have been developed. Each has advantages over the others which make it superior for solving certain types of problems.

A detailed User's Manual has been supplied along with this report and copies of the codes. The User's Manual is the guide on how to use each of the codes. It gives example problems, examples of the necessary computer input, aids for selection of this information, examples of the output, and its analysis.

5.2 Recommendations

The present NLGP algorithm and computer codes have limitations which have been presented in previous sections. There are certain areas in which future research efforts should be focused:

(1) Examination and development of other heuristic techniques to solve 0-1 nonlinear goal programming problems.

(2) Development of methods to solve mixed-integer NLGP problems.

(3) Development of generalized sensitivity analyses for NLGP and implementation of these methods into computer packages.

(4) Analysis of these and future codes using actual design problems with over 1,000 variables and over 1,000 objectives. This will greatly enhance future validation efforts.

(5) Including and testing future additional program modifications as they are developed.

APPENDIX A

NLGP/MPS-RS CODE: VERSION 1

```
C ***********************************************************************
C        NONLINEAR GOAL PROGRAMMING--PATTERN SEARCH CODE
C ***********************************************************************
C
C            THIS IS THE NIGP/MPS-RS NAVY CODE
C            NONLINEAR GOAL PROGRAMMING/MODIFIED PATTERN SEARCH CODE
C            WITH A RESOLUTION RIDGE SEARCH TECHNIQUE
C            VERSION 1: SCALAR STEP SIZE (EPS)
C                 --- JULY 15, 1980 ---
C
C
C                 ***CODE SPECIFICATIONS***
C                    2500 OBJECTIVES
C                    2500 VARIABLES
C                      10 PRIORITY LEVELS
C
C                 ***VARIABLES AND PARAMETERS***
C     ***NAME***          ***DESCRIPTION***
C        NOBJ                NUMBER OF OBJECTIVES
C        NPRIOR              NUMBER OF PRIORITY LEVELS
C        NVAR                NUMBER OF DECISION VARIABLES
C        NMAX                MAXIMUM NO. OF PATTERN SEARCH CYCLES ALLOWED
C        NCYCLE              COUNTER FOR PATTERN SEARCH CYCLES
C        X                   DECISION VARIABLE VECTOR USED FOR PERTURBATION
C        XBASE               BASE POINT VECTOR FOR PATTERN SEARCH
C        A                   ACHIEVEMENT VECTOR AT TEST POINT
C        ABEST               ACHIEVEMENT VECTOR FOR BEST SOLUTION
C        APOS,ANEG           TEMPORARY ACHIEVEMENT VECTORS FOR RIDGE SEACH
C        EPS,IPS             SCALARS OF PERTURBATION STEP SIZES USED FOR
C                            EXPLORATORY MOVES
C        LBD,UBD             LOWER AND UPPER BOUND VECTORS FOR DECISION VAR
C        OBJ                 OBJECTIVE FUNCTION VECTOR
C        RHS                 RIGHT-HAND SIDE VALUE VECTOR FOR OBJECTIVE FNS
C        N,P                 NEGATIVE AND POSITIVE DEVIATION VARIABLE VECT.
C        EPSY                TERMINATION TOLERANCE TEST VECTOR
C        ACCEL               ACCELERATION FACTOR
C        REDUCE              STEP SIZE REDUCTION FACTOR
C        DELTA               MINIMUM ALLOWABLE STEP SIZE
C        IMPROV              TEST SWITCH---IF IMPROV=1, X IMPROVES THE
C                            SOLUTION; IF IMPROV=0, X DOES NOT IMPROVE THE
C                            SOLUTION
C        IPRINT              OUTPUT SWITCH---IF IPRINT=1, RESULTS ARE
C                            PRINTED OUT AT EVERY PATTERN SEARCH CYCLE;
C                            IF IPRINT=0, ONLY THE FINAL RESULTS ARE OUTPUT
C        ITERM               TERMINATION TEST SWITCH---IF ITERM=1, THE
C                            SOLUTION IS TESTED FOR TERMINATION; IF ITERM=0
C                            ONLY A COMPARISON TEST IS PERFORMED
C        IPROG               COUNTER FOR IMPROVEMENTS DURING THE EXPLORA-
C                            TORY MOVES
C        IACCEL              INDICATOR FOR ACCELERATION MOVES:
C                            IACCEL = 1, ACCELERATION MOVE WAS SUCCESSFUL
C                            IACCEL =-1, ACCELERATION MOVE WAS UNSUCCESSFUL
C                            IACCEL = 0, ACCELERATION MOVE NOT ATTEMPTED
C        SAVE                A TEMPORARY SWITCHING INDICATOR FOR PATTERN
C                            AND RIDGE MOVES
C
C
C                  ***DATA INPUT GUIDE***
C
C--------CARD 1                          FORMAT
```

```
C         NOBJ                              IS
C         NPRIOR                            IS
C         NVAR                              IS
C         NMAX                              IS
C         IPRINT                            IS
C--------CARD 2---INITIAL POINT
C         X(K), K=1,NVAR                    8F10.0
C--------CARD 3---LOWER BOUNDS
C         LBD(K), K=1,NVAR                  8F10.0
C--------CARD 4---UPPER BOUNDS
C         UBD(K), K=1,NVAR                  8F10.0
C--------CARD 5---RIGHT-HAND SIDES
C         RHS(I), I=1,NOBJ                  8F10.0
C--------CARD 6---TOLERANCES FOR TERMINATION
C         EPSY(J),J=1,NPRIOR                8F10.0
C--------CARD 7
C         EPS---INITIAL STEP SIZE            F10.0
C         ACCEL---ACCELERATION FACTOR        F10.0
C         REDUCE---STEP SIZE REDUCTION
C                  FACTOR                    F10.0
C         DELTA---STEP SIZE TEST FACTOR      F10.0
C
C
C                  ***SUBROUTINES***
C         SUBROUTINES OBJFUN AND ACHFUN MUST BE USER SUPPLIED
C         FOR EACH NEW NLGP PROBLEM
C         SUBROUTINE OBJFUN CONTAINS THE OBJECTIVE FUNCTIONS
C         SUBROUTINE ACHFUN CONTAINS THE ACHIEVEMENT
C         FUNCTIONS, ONE FOR EACH PRIORITY LEVEL
C
C
C                  ***MAIN PROGRAM***
C
      REAL*4 IPS,LBD,N
      INTEGER*2 IMPROV,IPRINT,ITERM,IPROG,IACCEL
      COMMON/COMM01/NOBJ,NPRIOR,NVAR,NMAX,NCYCLE
      COMMON/COMM02/X(2500),XBASE(2500)
      COMMON/COMM03/A(10),ABEST(10)
      COMMON/COMM04/ACCEL,REDUCE,DELTA,IMPROV,IPRINT,ITERM
      COMMON/COMM05/LBD(2500),UBD(2500),EPS,IPS
      COMMON/COMM06/OBJ(2500),RHS(2500)
      COMMON/COMM07/N(2500),P(2500)
      COMMON/COMM08/EPSY(10)
      COMMON/COMM09/SAVE
C
C        BEGIN THE PROGRAM
      CALL DATAIN
C        ALL NECESSARY DATA IS INPUT AND PRINTED OUT
      CALL HJALG
C        BEGIN THE PATTERN SEARCH ALGORITHM
C        ALL FURTHER WORK IS COMPLETED IN THE SUBROUTINES
C        OF THE CODE
      STOP
      END
C ***************************************************************************
      SUBROUTINE DATAIN
C ***************************************************************************
C
C        SUBROUTINE DATAIN READS IN ALL THE NECESSARY DATA
C        AND PRINTS IT OUT AS A MEANS OF CHECKING IT.
```

```
C
C
      INTEGER*2 IMPRCV,IPRINT,ITERM,IPROG,IACCEL
      REAL*4 IPS,LBD
      COMMON/COMM01/NOBJ,NPRIOR,NVAR,NMAX,NCYCLE
      COMMON/COMM02/X(2500),XBASE(2500)
      COMMON/COMM04/ACCEL,REDUCE,DELTA,IMPRCV,IPRINT,ITERM
      COMMON/COMM05/LBD(2500),UBD(2500),EPS,IPS
      COMMON/COMM06/OBJ(2500),RHS(2500)
      COMMON/COMM08/EPSY(10)
C        READ ALL THE NECESSARY DATA
      READ 170,NOBJ,NPRIOR,NVAR,NMAX,IPRINT
      READ 175,(X(K),K=1,NVAR)
      READ 175,(LBD(K),K=1,NVAR)
      READ 175,(UBD(K),K=1,NVAR)
      READ 175,(RHS(I),I=1,NOBJ)
      READ 175,(EPSY(J),J=1,NPRIOR)
      READ 175,EPS,ACCEL,REDUCE,DELTA
C        PRINT AN ECHO CHECK OF THE DATA
  100 PRINT 180,NOBJ,NPRIOR,NVAR,NMAX,EPS,ACCEL,REDUCE,DELTA
      PRINT 181
      DO 110 K=1,NVAR
  110 PRINT 182,K,X(K),K,LBD(K),K,UBD(K)
      PRINT 183
      DO 120 I=1,NOBJ
  120 PRINT 184,I,RHS(I)
      PRINT 185
      DO 130 J=1,NPRIOR
  130 PRINT 186,J,EPSY(J)
      PRINT 187
      IF(IPRINT .EQ. 1) GO TO 140
      PRINT 188
      RETURN
  140 PRINT 189
  170 FORMAT(16I5)
  175 FORMAT(8F10.0)
  180 FORMAT('-',25X,'NUMBER OF OBJECTIVES =',I57,I5,/,' ',25X,'NUMBER O
     $F PRIORITIES =',I57,I5,/,' ',25X,'NUMBER OF DECISION VARIABLES =',
     $I57,I5,/,' ',25X,'MAXIMUM NUMBER OF ALLOWABLE',/,' ',29X,'PATTERN
     $SEARCH CYCLES =',I57,I5,/,' ',25X,'INITIAL STEP  SIZES =',I59,F10.
     $6,/,' ',25X,'ACCELERATION FACTOR =',I59,F10.6,/,' ',25X,'STEP SIZE
     $REDUCTION FACTOR =',I59,F10.6,/,' ',25X,'MINIMUM ALLOWABLE STEP SI
     $ZE =',I59,F10.6)
  181 FORMAT('-',25X,'VECTOR OF INITIAL ESTIMATES FOR DECISION VARIABLES
     $---X',/,' ',25X,'LOWER AND UPPER BOUNDS ON DECISION VARIABLES---LB
     $D AND UBD')
  182 FORMAT(' ',10X,'X(',I5,')  =  ',F10.5,7X,'LBD(',I5,')  =  ',F10.5,
     $7X,'UBD(',I5,')  =  ',F10.5)
  183 FORMAT('-',25X,'RIGHT-HAND SIDE VALUES FOR OBJECTIVE FUNCTIONS---R
     $HS')
  184 FORMAT(' ',25X,'RHS(',I5,')  =  ',F20.4)
  185 FORMAT('-',25X,'VECTOR OF ACHIEVEMENT FUNCTION TOLERANCES---EPSY')
  186 FORMAT(' ',25X,'EPSY(',I2,')  =  ',F10.6)
  187 FORMAT('-',25X,'ALL DATA HAS BEEN INPUT---BEGIN THE PATTERN SEARCH
     $')
  188 FORMAT(' ',25X,'NOTE: IPRINT = 0, SO ONLY THE FINAL SOLUTION WILL
     $BE PRINTED OUT')
  189 FORMAT(' ',25X,'NOTE: IPRINT = 1, SO THE SOLUTION WILL BE PRINTED
     $OUT AT EVERY PATTERN SEARCH CYCLE')
      RETURN
```

```
      END
C ***************************************************************************
      SUBROUTINE HJALG
C ***************************************************************************
C
C        SUBROUTINE HJALG IS THE HOOKE-JEEVES PATTERN
C        SEARCH ALGORITHM.  IT PERFORMS EXPLORATORY AND
C        PATTERN MOVES TO FIND THE 'BEST' SOLUTION TO THE
C        NONLINEAR GOAL PROGRAMMING PROBLEM.
C
C
      REAL*4 IPS,LBD
      INTEGER*2 IMPROV,IPRINT,ITERM,IPROG,IACCEL
      COMMON/COMM01/NOBJ,NPRIOR,NVAR,NMAX,NCYCLE
      COMMON/COMM02/X(2500),XBASE(2500)
      COMMON/COMM03/A(10),ABEST(10)
      COMMON/COMM04/ACCEL,REDUC,DELTA,IMPROV,IPRINT,ITERM
      COMMON/COMM05/LBD(2500),UBD(2500),EPS,IPS
      COMMON/COMM09/SAVE
      CALL ACHFUN
C        SUBROUTINE ACHFUN FORMS THE ACHIEVEMENT VECTOR FOR
C        THE INITIAL BASE POINT.  IT IS TESTED FOR ALL ZERO
C        VALUES---IF SO, THE INITIAL POINT IS OPTIMAL.
C        OTHERWISE, IT IS SET AS THE BEST VALUE AND IS PRINTED.
      DO 100 J=1,NPRIOR
      IF(A(J) .NE. 0.0)GO TO 203
  100 CONTINUE
C        ALL PRIORITY LEVELS ARE SATISFIED AT ZERO VALUE.
C        THE INITIAL POINT IS OPTIMAL---TERMINATE.
  200 PRINT 290
      DO 201 J=1,NPRIOR
  201 PRINT 282,J,A(J)
      PRINT 291
      DO 202 K=1,NVAR
  202 PRINT 280,K,X(K)
      GO TO 295
  203 PRINT 281
      DO 204 J=1,NPRIOR
      ABEST(J) = A(J)
  204 PRINT 282,J,A(J)
C        ** INITIALIZATION OF PATTERN SEARCH PARAMETERS **
C        NCYCLE COUNTS THE NUMBER OF PATTERN SEARCHES USED
C        IACCEL IS AN INDICATOR WHICH TELLS IF THE EXPLORATORY
C        MOVES ARE MADE ABOUT AN ACCELERATION POINT
      NCYCLE = 0
      IACCEL = 0
C        SET THE INITIAL BASE POINT AND THE INITIAL STEP
C        SIZE BEFORE THE EXPLORATORY MOVES
      DO 205 K=1,NVAR
  205 XBASE(K) = X(K)
      IPS = EPS
C        START THE PATTERN SEARCH
C        ** EXPLORATORY MOVES **
  206 IF(NCYCLE .GT. NMAX) GO TO 260
      IPROG = 0
C        IPROG COUNTS THE NUMBER OF IMPROVEMENTS DURING THE
C        EXPLORATORY MOVES.  IT IS USED TO DECIDE IF A PATTERN
C        MOVE SHOULD BE PERFORMED.
  207 DO 210 K=1,NVAR
C        PERTURBATIONS IN THE POSITIVE DIRECTION
```

```
C        TEST STEP SIZE COMPONENT.  IF LESS THAN THE MINIMUM
C        ALLOWABLE STEP SIZE, A RIDGE SEARCH WILL BE PERFORMED
      IF(IPS .LT. DELTA) GO TO 255
      X(K) = X(K) + IPS
      IF(X(K) .LT. LBD(K))GO TO 208
      IF(X(K) .GT. UBD(K)) GO TO 208
      CALL ACHFUN
C        ACHFUN FORMS THE ACHIEVEMENT VECTOR AT THE TEST POINT
      ITERM = 0
      CALL DECIDE
C        DECIDE COMPARES THE ACHIEVEMENT VECTORS OF THE
C        BASE POINT AND THE TEST POINT.  IF X(K) IMPROVES THE
C        SOLUTION, IT BECOMES A TEMPORARY HEAD POINT AND
C        EXPLORATORY MOVES ARE CONTINUED FROM IT.  IF THERE
C        IS NO IMPROVEMENT, THEN EXPLORE IN THE NEGATIVE DIRECTION
      IF(IMPROV .EQ. 0) GO TO 208
      IPROG = IPROG + 1
      GO TO 210
C        PERTURBATIONS IN THE NEGATIVE DIRECTION
  208 X(K) = X(K) - 2.0*IPS
      IF(X(K) .LT. LBD(K)) GO TO 209
      IF(X(K) .GT. UBD(K)) GO TO 209
      CALL ACHFUN
C        ACHFUN FORMS THE ACHIEVEMENT VECTOR AT THE TEST POINT
      ITERM = 0
      CALL DECIDE
C        IF X(K) IMPROVES THE SOLUTION, IT BECOMES THE
C        TEMPORARY HEAD POINT AND THE NEXT VARIABLE IS
C        PERTURBED.  IF THERE IS NO IMPROVEMENT, XBASE(K)
C        REMAINS THE TEMPORARY HEAD POINT.
      IF(IMPROV .EQ.0) GO TO 209
      IPROG = IPROG + 1
      GO TO 210
  209 X(K) = X(K) + IPS
  210 CONTINUE
      NCYCLE = NCYCLE + 1
C        THE EXPLORATORY MOVE DO LOOP HAS CONSTRUCTED A
C        NEW POINT.  TEST THE ACHIEVEMENT VECTOR
C        FOR ALL ZERO VALUES
      DO 211 J=1,NPRIOR
      IF(A(J) .NE.0.000) GO TO 212
  211 CONTINUE
      GO TO 200
C        THE ACHIEVEMENT VECTOR IS NOT ALL ZEROES
C        TEST THE POINT FOR IMPROVEMENT.  IF IT GIVES
C        IMPROVEMENT, A PATTERN MOVE IS PERFORMED.  IF NOT
C        THEN STEP SIZES WILL BE REDUCED AND NEW EXPLORATORY
C        MOVES WILL BE PERFORMED.
  212 IF(IPROG .GT. 0) GO TO 214
      IF(IACCEL)250,213,215
C        THE EXPLORATORY MOVE HAS BEEN UNSUCCESSFUL.  STEP
C        SIZES WILL BE REDUCED AND NEW EXPLORATORY MOVES
C        WILL BEGIN.
  213 PRINT 283
C        ** REDUCE STEP SIZES **
      IPS = IPS * REDUCE
      GO TO 206
C        THE EXPLORATORY MOVE HAS BEEN SUCCESSFUL.
C        A PATTERN MOVE WILL BE PERFORMED
  214 PRINT 279
```

```
C        ** ACCELERATION **
  215 DO 220 K=1,NVAR
      SAVE = X(K)
      X(K) = X(K) + ACCEL * (X(K) - XBASE(K))
      IF(X(K) .LT.LBD(K))X(K) = LBD(K)
      IF(X(K) .LE. UBD(K))GO TO 220
      X(K) = UBD(K)
  220 XBASE(K) = SAVE
C        TEST THE ACHIEVEMENT VECTOR AT THE ACCELERATION POINT
      CALL ACHFUN
      ITERM = 1
      CALL DECIDE
      IF(IMPROV .EQ. 1) GO TO 225
      IACCEL = -1
      GO TO 230
  225 IACCEL = 1
      GO TO 240
C        THE ACCELERATION PT. GIVES NO IMPROVEMENT.
C        EXPLORATORY MOVES WILL BE PERFORMED ABOUT THE
C        ACCELERATION PT. TO FIND IMPROVEMENT.
  230 PRINT 286
  231 IF(IPRINT .EQ.0) GO TO 206
      PRINT 287,NCYCLE
      DO 232 K=1,NVAR
  232 PRINT 280,K,XBASE(K)
      PRINT 288
      DO 235 J=1,NPRIOR
  235 PRINT 282,J,ABEST(J)
      GO TO 206
C        THE ACCELERATION POINT GIVES IMPROVEMENT.
C        EXPLORE ABOUT IT FOR IMPROVEMENT.
  240 PRINT 285
      IF(IPRINT .EQ. 0)GO TO 206
      PRINT 287,NCYCLE
      DO 241 K=1,NVAR
  241 PRINT 280,K,X(K)
      PRINT 288
      DO 245 J=1,NPRIOR
  245 PRINT 282,J,A(J)
      GO TO 206
C        EXPLORATION AROUND THE ACCELERATION POINT
C        DOES NOT YIELD ANY IMPROVEMENT
C        RESET THE BASE POINT AND GO TO EXPLORATORY MOVES.
  250 IF(IACCEL .EQ. 0) GO TO 255
  251 DO 252 K=1,NVAR
  252 X(K) = XBASE(K)
      IACCEL = 0
      GO TO 206
C        STEP SIZE REDUCTIONS EQUALLED.
C        A RESOLUTION RIDGE SEARCH WILL BE PERFORMED
  255 PRINT 284
  259 CALL RIDGE
C        SUBROUTINE RIDGE WILL TRY TO FIND A NEW BASE POINT
C        IN AN OBLIQUE DIRECTION TO THE PRESENT EXPLORATORY
C        SEARCH DIRECTION.  IF SUCCESSFUL, THE PATTERN
C        SEARCH WILL BEGIN AT THIS NEW POINT.  IF UNSUCCESSFUL,
C        THE ALGORITHM WILL BE TERMINATED.
      IACCEL = 0
      GO TO 206
C        THE MAXIMUM NUMBER OF PATTERN SEARCH CYCLES HAS BEEN
```

```
C        EXCEEDED.  THE BEST VALUES OF THE VARIABLES AND THE
C        ACHIEVEMENT VECTOR ARE PRINTED OUT, AND THE ALGORITHM
C        WILL BE TERMINATED.
  260 PRINT 289
      IF(IACCEL .LT. 1) GO TO 265
      DO 261 K=1,NVAR
  261 PRINT 280,K,X(K)
      PRINT 288
      DO 262 J=1,NPRIOR
  262 PRINT 282,J,A(J)
      GO TO 295
  265 DO 266 K=1,NVAR
  266 PRINT 280,K,XBASE(K)
      PRINT 288
      DO 267 J=1,NPRIOR
  267 PRINT 282,J,ABEST(J)
  270 GO TO 295
  279 FORMAT('-',25X,'THE EXPLORATORY MOVE HAS BEEN SUCCESSFUL',/,' ',25
     $X,'A PATTERN MOVE WILL BE PERFORMED')
  280 FORMAT(' ',34X,'X(',I5,')  =  ',F13.6)
  281 FORMAT(' ',25X,'INITIAL ACHIEVEMENT VECTOR VALUES')
  282 FORMAT(' ',34X,'A(',I2,')  =  ',F13.6)
  283 FORMAT('-',25X,'TEMPORARY BASE POINT DOES NOT IMPROVE THE SOLUTION
     $',/,' ',25X,'THE STEP SIZES WILL BE REDUCED',/,' ',25X,'NEW EXPLOR
     $ATORY MOVES WILL BE PERFORMED ABOUT XBASE')
  284 FORMAT(' ',25X,'MAXIMUM STEP SIZE REDUCTIONS EXCEEDED FOR THIS EXP
     $LORATION',/,' ',25X,'A RIDGE SEARCH WILL BE PERFORMED')
  285 FORMAT('-',25X,'THE ACCELERATION POINT YIELDS AN IMPROVED SOLUTION
     $',/,' ',25X,'EXPLORATORY MOVES WILL BEGIN AT THIS NEW POINT')
  286 FORMAT('-',25X,'THE ACCELERATION POINT DOES NOT YIELD AN IMPROVED
     $SOLUTION',/,' ',25X,'EXPLORATORY MOVES WILL BEGIN AT THIS POINT')
  287 FORMAT('-',25X,'THE BEST SOLUTION AT PATTERN SEARCH NUMBER ',I3,'
     $FOLLOWS')
  288 FORMAT(' ',25X,'THE ACHIEVEMENT VECTOR VALUES ARE:')
  289 FORMAT(' ',25X,'MAXIMUM NUMBER OF PATTERN SEARCH CYCLES EXCEEDED',
     $/,' ',25X,'THE ALGORITHM IS TERMINATED',/,' ',25X,'THE BEST SOLUTI
     $ON TO THIS POINT FOLLOWS')
  290 FORMAT('-',20X,43(1H*),/,' ',25X,'OPTIMAL ACHIEVEMENT VECTOR VALUE
     $S',/,' ',20X,43(1H*))
  291 FORMAT('-',20X,43(1H*),/,' ',25X,'OPTIMAL DECISION VARIABLE VALUES
     $',/,' ',20X,43(1H*))
  295 STOP
      END
C ***************************************************************
      SUBROUTINE DEVVAR
C ***************************************************************
C
C        SUBROUTINE DEVVAR CALCULATES THE POSITIVE AND
C        NEGATIVE DEVIATION VARIABLES FOR EACH OBJECTIVE
C        FUNCTION USING THE PREVIOUSLY EVALUATED OBJECTIVE
C        FUNCTIONS AND THE RIGHT-HAND SIDE VALUES.
C
C
      REAL*4 N
      COMMON/COMM01/NOBJ,NPRIOR,NVAR,NMAX,NCYCLE
      COMMON/COMM02/X(2500),XBASE(2500)
      COMMON/COMM06/OBJ(2500),RHS(2500)
      COMMON/COMM07/N(2500),P(2500)
      CALL OBJFUN
  400 DO 430 I=1,NOBJ
```

```
      IF(OBJ(I) - RHS(I))410,420,420
  410 N(I) = RHS(I) - OBJ(I)
      P(I) = 0.0
      GO TO 430
  420 N(I) = 0.00
      P(I) = OBJ(I) - RHS(I)
  430 CONTINUE
      RETURN
      END
C *************************************************************************
      SUBROUTINE DECIDE
C *************************************************************************
C
C        SUBROUTINE DECIDE IS USED FOR TWO PURPOSES:
C           (1) TO COMPARE THE ACHIEVEMENT VECTOR VALUES OF
C        THE BASE POINT AND A TEST POINT DURING THE EXPLORATORY
C        MOVES AND SELECT THE BEST POINT FOR FUTURE MOVES
C           (2) TO TEST AN ACHIEVEMENT VECTOR FOR TERMINATION
C        OF THE ALGORITHM.  THIS CAN OCCUR IN TWO METHODS:
C             ****TERMINATION BY MEANS OF ZERO FOR ALL
C              PRIORITY LEVELS, AND
C             ****TERMINATION BY MEETING SPECIFIED TOLERANCES
C
C        THE PARAMETER ITERM IS AN INDICATOR FOR TERMINATION
C        TESTING.  IF ITERM = 0, ONLY THE COMPARISON TEST IS
C        PERFORMED.  IF ITERM = 1, THE TERMINATION TESTS WILL
C        ALSO BE PERFORMED.
C
C
      INTEGER*2 IMPROV,IPRINT,ITERM
      COMMON/COMM01/NOBJ,NPRIOR,NVAR,NMAX,NCYCLE
      COMMON/COMM02/X(2500),XBASE(2500)
      COMMON/COMM03/A(10),ABEST(10)
      COMMON/COMM04/ACCEL,REDUC,DELTA,IMPROV,IPRINT,ITERM
      COMMON/COMM08/EPSY(10)
C        ** COMPARISON OF ACHIEVEMENT VECTORS FOR EXPLORATORY MOVES **
  600 DO 610 J=1,NPRIOR
      IF(A(J) .GT. ABEST(J) +0.0001) GO TO 630
      IF(A(J) +0.0001 .LT. ABEST(J))GO TO 620
      IF(J .EQ. NPRIOR) GO TO 630
  610 CONTINUE
C        THE TEST POINT IMPROVES THE SOLUTION
  620 IMPROV = 1
      GO TO 640
C        THE TEST POINT DOES NOT IMPROVE THE SOLUTION
  630 IMPROV = 0
C        CHECK THE INDICATOR FOR FURTHER TERMINATION TESTING
  640 IF(ITERM .EQ. 0)GO TO 670
      CONTINUE
C        ** TERMINATION TESTS **
C        (1)TERMINATION TEST BY MEANS OF ZERO FOR ALL PRIORITIES
  650 DO 655 J=1,NPRIOR
      IF(A(J) .NE. 0.0)GO TO 660
  655 CONTINUE
C        ALL PRIORITY LEVELS ARE ZERO---THE OPTIMAL SOLUTION
C        HAS BEEN FOUND---TERMINATE THE ALGORITHM
      GO TO 680
C        (2)TERMINATION TEST BY TOLERANCES
  660 DO 665 J=1,NPRIOR
      IF(ABS(A(J) - ABEST(J)) .GT. EPSY(J))GO TO 670
```

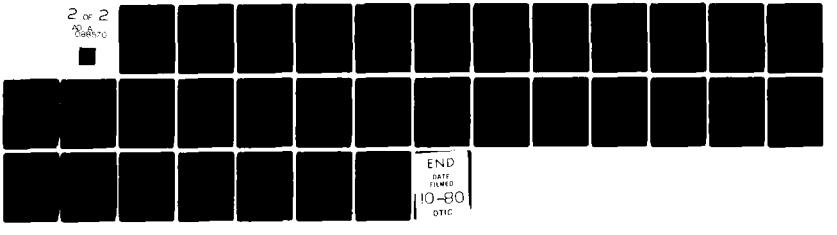
```
  665 CONTINUE
C         ALL TOLERANCES ARE SATISFIED---A SOLUTION HAS BEEN
C         FOUND---THE SEARCH ALGORITHM IS COMPLETE
      GO TO 680
C         A TOLERANCE IS NOT SATISFIED---A BETTER SOLUTION
C         WILL BE SOUGHT---RETURN TO THE PATTERN SEARCH
  670 IF(IMPROV .EQ. 0) GO TO 695
      DO 675 J=1,NPRIOR
  675 ABEST(J) = A(J)
      GO TO 695
  680 PRINT 690
      PRINT 691
      DO 682 J=1,NPRIOR
  682 PRINT 692,J,A(J)
      PRINT 693
      DO 685 K=1,NVAR
  685 PRINT 694,K,X(K)
      STOP
  690 FORMAT('-',25X,'THE SEARCH ALGORITHM IS COMPLETE---ALL TOLERANCES
     $FOR A SOLUTION ARE SATISFIED')
  691 FORMAT('-',20X,43(1H*),/,' ',25X,'OPTIMAL ACHIEVEMENT VECTOR VALUE
     $S',/,' ',20X,43(1H*))
  692 FORMAT(' ',34X,'A(',I2,')  =  ',F13.6)
  693 FORMAT('-',20X,43(1H*),/,' ',25X,'OPTIMAL DECISION VARIABLE VALUES
     $',/,' ',20X,43(1H*))
  694 FORMAT(' ',34X,'X(',I5,')  =  ',F13.6)
  695 RETURN
      END
C *********************************************************************
      SUBROUTINE RIDGE
C *********************************************************************
C
C         SUBROUTINE RIDGE IS CALLED WHEN THE PATTERN SEARCH
C         CAN NO LONGER FIND PATTERN OR EXPLORATORY MOVES
C         WHICH IMPROVE THE ACHIEVEMENT VECTOR.  RIDGE
C         EVALUATES EXPLORATORY POINTS WHICH ARE IN OBLIQUE
C         DIRECTIONS TO THE USUAL EXPLORATION AXES.  RIDGE
C         ATTEMPTS TO FIND A RESOLUTION RIDGE, IF IT EXISTS,
C         AND MOVE THE PATTERN IN THAT DIRECTION.
C
C
      DIMENSION APOS(10),ANEG(10)
      REAL*4 IPS,LBD
      INTEGER*2 IMPROV,IPRINT,ITERM
      COMMON/COMM01/NOBJ,NPRIOR,NVAR,NMAX,NCYCLE
      COMMON/COMM02/X(2500),XBASE(2500)
      COMMON/COMM03/A(10),ABEST(10)
      COMMON/COMM04/ACCEL,REDUCE,DELTA,IMPROV,IPRINT,ITERM
      COMMON/COMM05/LBD(2500),UBD(2500),EPS,IPS
      COMMON/COMM09/SAVE
C         PERTURB EACH VARIABLE ONE AT A TIME
C         PERTURB IN THE POSITIVE DIRECTION
      DO 800 K=1,NVAR
  800 X(K)  = XBASE(K)
  801 DO 865 K=1,NVAR
      X(K)  = X(K)  + IPS
      IF(X(K) .LT. LBD(K)) GO TO 805
      IF(X(K) .GT. UBD(K)) GO TO 810
      GO TO 815
  805 X(K)  = LBD(K)
```

```
      GO TO 815
  810 X(K) = UBD(K)
  815 CALL ACHFUN
C        ACHFUN FORMS THE ACHIEVEMENT VECTOR IN THE POSITIVE
C        DIRECTION
      DO 820 J=1,NPRIOR
  820 APOS(J) = A(J)
C        PERTURB IN THE NEGATIVE DIRECTION
      X(K) = X(K) - 2.0 * IPS
      IF(X(K) .LT. LBD(K)) GO TO 825
      IF(X(K) .GT. UBD(K)) GO TO 830
      GO TO 835
  825 X(K) = LBD(K)
      GO TO 835
  830 X(K) = UBD(K)
  835 CALL ACHFUN
C        ACHFUN FORMS THE ACHIEVEMENT VECTOR IN THE NEGATIVE
C        DIRECTION
      DO 840 J=1,NPRIOR
  840 ANEG(J) = A(J)
C        COMPARE APOS(J) AND ANEG(J).  SELECT THE MINIMUM
C        VIA A STEPWISE PREEMPTIVE PROCEDURE.  THIS DETERMINES
C        A DIRECTION  FOR THE RIDGE POINT.
      DO 850 J=1,NPRIOR
      IF(APOS(J) .LT. ANEG(J))GO TO 855
      IF(APOS(J) .GT. ANEG(J))GO TO 860
      IF(J .EQ. NPRIOR)GO TO 855
  850 CONTINUE
  855 X(K) = XBASE(K) + IPS
      GO TO 861
  860 X(K) = XBASE(K) - IPS
C        RETURN X(K) TO ITS BASE POINT VALUE, XBASE, BEFORE
C        THE PERTURBATION OF THE NEXT VARIABLE
  861 SAVE = XBASE(K)
      XBASE(K) = X(K)
  865 X(K) = SAVE
C        THE NEW POINT, XRIDGE, HAS BEEN FORMED IN AN OBLIQUE
C        DIRECTION TO THE OLD EXPLORATORY MOVES.  ITS
C        ACHIEVEMENT VECTOR IS COMPUTED AND COMPARED TO THE
C        BEST VALUE AT POINT XBASE.
      DO 866 K=1,NVAR
      SAVE = XBASE(K)
      XBASE(K) = X(K)
  866 X(K) = SAVE
      CALL ACHFUN
      ITERM = 0
      CALL DECIDE
C        IF THE RIDGE POINT GIVES NO IMPROVEMENT, THE PRESENT
C        BASE POINT IS THE BEST SOLUTION, AND THE ALGORITHM
C        IS TERMINATED.
C        IF THE RIDGE POINT GIVES IMPROVEMENT, AN EVEN BETTER
C        POINT WILL BE SOUGHT ALONG THE RESOLUTION RIDGE.
      IF(IMPROV .EQ.1) GO TO 869
      CONTINUE
C        XBASE IS THE BEST SOLUTION
      PRINT 891
      PRINT 897
      DO 867 K=1,NVAR
  867 PRINT 892,K,XBASE(K)
      PRINT 898
```

AD-A088 570 PENNSYLVANIA STATE UNIV UNIVERSITY PARK DEPT OF INDU--ETC F/G 9/2
ADVANCES IN MULTICRITERIA ENGINEERING DESIGN: COMPUTATIONAL TEC--ETC(U)
JUL 80 T L ELCHAK, J P IGNIZIO, T YANG N00014-79-C-0730

UNCLASSIFIED NL

2 OF 2
AD A088570



END
DATE FILMED
10-80
DTIC

```
      DO 868 J=1,NPRIOR
  868 PRINT 893,J,ABEST(J)
      STOP
C        THE RIDGE POINT GIVES AN IMPROVED SOLUTION.
C        TRY TO ACCELERATE ALONG THE RESOLUTION RIDGE
  869 PRINT 894
  870 DO 871 K=1,NVAR
      SAVE = X(K)
      X(K) = X(K) + ACCEL * (X(K) - XBASE(K))
  871 XBASE(K) = SAVE
      CALL ACHFUN
      ITERM = 0
      CALL DECIDE
C        IF THE ACCELERATION POINT YIELDS AN IMPROVED SOLUTION,
C        TRY TO ACCELERATE FARTHER ALONG THE RESOLUTION RIDGE
C        IF THERE IS NO IMPROVEMENT, THE RIDGE POINT BECOMES
C        THE NEW POINT FOR THE RENEWED PATTERN SEARCH.
      IF(IMPROV .EQ. 0) GO TO 872
      PRINT 896
      GO TO 870
  872 PRINT 895
      PRINT 897
      DO 873 K=1,NVAR
  873 PRINT 892,K,XBASE(K)
      PRINT 898
      DO 875 J=1,NPRIOR
  875 PRINT 893,J,ABEST(J)
C        RESET THE BASE POINT AND STEP SIZE VECTORS AND
C        RETURN TO THE EXPLORATORY MOVES.
C        NO IMPROVEMENT FROM ACCELERATION POINT
      DO 877 K=1,NVAR
  877 X(K) = XBASE(K)
      EPS = EPS
      DO 879 J=1,NPRIOR
  879 A(J) = ABEST(J)
      GO TO 899
  891 FORMAT('-',25X,'THE RIDGE SEARCH IS COMPLETE',/,' ',25X,'NO IMPROV
     $ED SOLUTION CAN BE FOUND',/,' ',25X,'THE CURRENT BASE POINT IS THE
     $ BEST SOLUTION THAT CAN BE FOUND THROUGH THE PATTERN SEARCH')
  892 FORMAT(' ',34X,'X(',I5,') = ',F13.6)
  893 FORMAT(' ',34X,'A(',I2,')    = ',F13.6)
  894 FORMAT('-',25X,'THE RIDGE POINT GIVES AN IMPROVED SOLUTION',/,' ',
     $25X,'TRY TO ACCELERATE ALONG THE RESOLUTION RIDGE')
  895 FORMAT('-',25X,'ACCELERATION ALONG THE RESOLUTION RIDGE YIELDS NO
     $IMPROVED SOLUTION',/,' ',25X,'THE RIDGE POINT IS THE NEW BASE POIN
     $T')
  896 FORMAT('-',25X,'ACCELERATION ALONG THE RESOLUTION RIDGE YIELDS AN
     $IMPROVED SOLUTION',/,' ',25X,'TRY TO ACCELERATE FURTHER')
  897 FORMAT('-',25X,'DECISION VARIABLES')
  898 FORMAT('-',25X,'ACHIEVEMENT VALUES')
  899 RETURN
      END
C ***************************************************************************
      SUBROUTINE ACHFUN
C ***************************************************************************
C
      REAL*4 N
      COMMON/COMM01/NOBJ,NPRIOR,NVAR,NMAX,NCYCLE
      COMMON/COMM02/X(2500),XBASE(2500)
      COMMON/COMM03/A(10),ABEST(10)
```

```
      COMMON/COMM07/N(2500),P(2500)
      CALL DEVVAR
C        ACHIEVEMENT FUNCTIONS ARE ENTERED HERE.
      RETURN
      END
C ***************************************************************************
       SUBROUTINE OBJFUN
C ***************************************************************************
C
      COMMON/COMM01/NOBJ,NPRIOR,NVAR,NMAX,NCYCLE
      COMMON/COMM02/X(2500),XBASE(2500)
      COMMON/COMM06/OBJ(2500),RHS(2500)
C        OBJECTIVE FUNCTIONS ARE ENTERED HERE
      RETURN
      END
```

APPENDIX B

NLGP/MPS-RS CODE: VERSION 2

```
C ***********************************************************************
C       NONLINEAR GOAL PROGRAMMING--PATTERN SEARCH CODE
C ***********************************************************************
C
C           THIS IS THE NLGP/MPS-RS NAVY CODE
C           NONLINEAR GOAL PROGRAMMING/MODIFIED PATTERN SEARCH CODE
C           WITH A RESOLUTION RIDGE SEARCH TECHNIQUE
C           VERSION 2: VECTOR STEP SIZES (EPS)
C                --- JULY 15, 1980 ---
C
C
C                ***CODE SPECIFICATIONS***
C                   2500 OBJECTIVES
C                   2500 VARIABLES
C                     10 PRIORITY LEVELS
C
C                ***VARIABLES AND PARAMETERS***
C     ***NAME***          ***DESCRIPTION***
C        NOBJ               NUMBER OF OBJECTIVES
C        NPRIOR             NUMBER OF PRIORITY LEVELS
C        NVAR               NUMBER OF DECISION VARIABLES
C        NMAX               MAXIMUM NO. OF PATTERN SEARCH CYCLES ALLOWED
C        NCYCLE             COUNTER FOR PATTERN SEARCH CYCLES
C        X                  DECISION VARIABLE VECTOR USED FOR PERTURBATION
C        XBASE              BASE POINT VECTOR FOR PATTERN SEARCH
C        A                  ACHIEVEMENT VECTOR AT TEST POINT
C        ABEST              ACHIEVEMENT VECTOR FOR BEST SOLUTION
C        APOS,ANEG          TEMPORARY ACHIEVEMENT VECTORS FOR RIDGE SEACH
C        EPS,IPS            PERTURBATION STEP SIZE VECTORS USED FOR FOR
C                           EXPLORATORY MOVES
C        LBD,UBD            LOWER AND UPPER BOUND VECTORS FOR DECISION VAR
C        OBJ                OBJECTIVE FUNCTION VECTOR
C        RHS                RIGHT-HAND SIDE VALUE VECTOR FOR OBJECTIVE FNS
C        N,P                NEGATIVE AND POSITIVE DEVIATION VARIABLE VECT.
C        EPSY               TERMINATION TOLERANCE TEST VECTOR
C        ACCEL              ACCELERATION FACTOR
C        REDUCE             STEP SIZE REDUCTION FACTOR
C        DELTA              MINIMUM ALLOWABLE STEP SIZE
C        IMPROV             TEST SWITCH---IF IMPROV=1, X IMPROVES THE
C                           SOLUTION; IF IMPROV=0, X DOES NOT IMPROVE THE
C                           SOLUTION
C        IPRINT             OUTPUT SWITCH---IF IPRINT=1, RESULTS ARE
C                           PRINTED OUT AT EVERY PATTERN SEARCH CYCLE;
C                           IF IPRINT=0, ONLY THE FINAL RESULTS ARE OUTPUT
C        ITERM              TERMINATION TEST SWITCH---IF ITERM=1, THE
C                           SOLUTION IS TESTED FOR TERMINATION; IF ITERM=0
C                           ONLY A COMPARISON TEST IS PERFORMED
C        IPROG              COUNTER FOR IMPROVEMENTS DURING THE EXPLORA-
C                           TORY MOVES
C        IACCEL             INDICATOR FOR ACCELERATION MOVES:
C                           IACCEL = 1, ACCELERATION MOVE WAS SUCCESSFUL
C                           IACCEL =-1, ACCELERATION MOVE WAS UNSUCCESSFUL
C                           IACCEL = 0, ACCELERATION MOVE NOT ATTEMPTED
C        SAVE               A TEMPORARY SWITCHING INDICATOR FOR PATTERN
C                           AND RIDGE MOVES
C
C
C                  ***DATA INPUT GUIDE***
```

```
C
C--------CARD 1                              FORMAT
C        NOBJ                               I5
C        NPRIOR                             I5
C        NVAR                               I5
C        NMAX                               I5
C        IPRINT                             I5
C--------CARD 2---INITIAL POINT
C        X(K), K=1,NVAR                     8F10.0
C--------CARD 3---PERTURBATION STEP SIZES
C        EPS(K),K=1,NVAR                    8F10.0
C--------CARD 4---LOWER BOUNDS
C        LBD(K), K=1,NVAR                   8F10.0
C--------CARD 5---UPPER BOUNDS
C        UBD(K), K=1,NVAR                   8F10.0
C--------CARD 6---RIGHT-HAND SIDES
C        RHS(I), I=1,NOBJ                   8F10.0
C--------CARD 7---TOLERANCES FOR TERMINATION
C        EPSY(J),J=1,NPRIOR                 8F10.0
C--------CARD 8
C        ACCEL---ACCELERATION FACTOR         F10.0
C        REDUCE---STEP SIZE REDUCTION
C                FACTOR                      F10.0
C        DELTA---STEP SIZE TEST FACTOR       F10.0
C
C
C                ***SUBROUTINES***
C        SUBROUTINES OBJFUN AND ACHFUN MUST BE USER SUPPLIED
C        FOR EACH NEW NLGP PROBLEM
C        SUBROUTINE OBJFUN CONTAINS THE OBJECTIVE FUNCTIONS
C        SUBROUTINE ACHFUN CONTAINS THE ACHIEVEMENT
C        FUNCTIONS, ONE FOR EACH PRIORITY LEVEL
C
C
C                ***MAIN PROGRAM***
C
      REAL*4 IPS,LBD,N
      INTEGER*2 IMPROV,IPRINT,ITERM,IPROG,IACCEL
      COMMON/COMM01/NOBJ,NPRIOR,NVAR,NMAX,NCYCLE
      COMMON/COMM02/X(2500),XBASE(2500)
      COMMON/COMM03/A(10),ABEST(10)
      COMMON/COMM04/ACCEL,REDUCE,DELTA,IMPROV,IPRINT,ITERM
      COMMON/COMM05/LBD(2500),UBD(2500),EPS(2500),IPS(2500)
      COMMON/COMM06/OBJ(2500),RHS(2500)
      COMMON/COMM07/N(2500),P(2500)
      COMMON/COMM08/EPSY(10)
      COMMON/COMM09/SAVE
C
C        BEGIN THE PROGRAM
      CALL DATAIN
C        ALL NECESSARY DATA IS INPUT AND PRINTED OUT
      CALL HJAIG
C        BEGIN THE PATTERN SEARCH ALGORITHM
C        ALL FURTHER WORK IS COMPLETED IN THE SUBROUTINES
C        OF THE CODE
      STOP
      END
C ***************************************************************************
```

```
      SUBROUTINE DATAIN
C ***************************************************************************
C
C        SUBROUTINE DATAIN READS IN ALL THE NECESSARY DATA
C        AND PRINTS IT OUT AS A MEANS OF CHECKING IT.
C
C
      INTEGER*2 IMPROV,IPRINT,ITERM,IPROG,IACCEL
      REAL*4 IPS,LBD
      COMMON/COMM01/NOBJ,NPRIOR,NVAR,NMAX,NCYCLE
      COMMON/COMM02/X(2500),XBASE(2500)
      COMMON/COMM04/ACCEL,REDUCE,DELTA,IMPROV,IPRINT,ITERM
      COMMON/COMM05/LBD(2500),UBD(2500),EPS(2500),IPS(2500)
      COMMON/COMM06/OBJ(2500),RHS(2500)
      COMMON/COMM08/EPSY(10)
C        READ ALL THE NECESSARY DATA
      READ 170,NOBJ,NPRIOR,NVAR,NMAX,IPRINT
      READ 175,(X(K),K=1,NVAR)
      READ 175,(EPS(K),K=1,NVAR)
      READ 175,(LBD(K),K=1,NVAR)
      READ 175,(UBD(K),K=1,NVAR)
      READ 175,(RHS(I),I=1,NOBJ)
      READ 175,(EPSY(J),J=1,NPRIOR)
      READ 175,ACCEL,REDUCE,DELTA
C        PRINT AN ECHO CHECK OF THE DATA
  100 PRINT 180,NOBJ,NPRIOR,NVAR,NMAX,ACCEL,REDUCE,DELTA
      PRINT 181
      DO 110 K=1,NVAR
  110 PRINT 182,K,X(K),K,EPS(K),K,LBD(K),K,UBD(K)
      PRINT 183
      DO 120 I=1,NOBJ
  120 PRINT 184,I,RHS(I)
      PRINT 185
      DO 130 J=1,NPRIOR
  130 PRINT 186,J,EPSY(J)
      PRINT 187
      IF(IPRINT .EQ. 1) GO TO 140
      PRINT 188
      RETURN
  140 PRINT 189
  170 FORMAT(16I5)
  175 FORMAT(8F10.0)
  180 FORMAT('-',25X,'NUMBER OF OBJECTIVES =',T57,I5,/,' ',25X,'NUMBER O
     $F PRIORITIES =',T57,I5,/,' ',25X,'NUMBER OF DECISION VARIABLES =',
     $T57,I5,/,' ',25X,'MAXIMUM NUMBER OF ALLOWABLE',/,' ',29X,'PATTERN
     $SEARCH CYCLES =',T57,I5,/,' ',25X,'ACCELERATION FACTOR =',T59,F10.
     $6,/,' ',25X,'STEP SIZE REDUCTION FACTOR =',T59,F10.6,/,' ',25X,'MI
     $NIMUM ALLOWABLE STEP SIZE =',T59,F10.6)
  181 FORMAT('-',25X,'VECTOR OF INITIAL ESTIMATES FOR DECISION VARIABLES
     $---X',/,' ',25X,'PERTURBATION STEP SIZES FOR THESE VARIABLES---EPS
     $',/,' ',25X,'LOWER AND UPPER BOUNDS FOR DECISION VARIABLES---LBD A
     $ND UBD')
  182 FORMAT(' ',10X,'X(',I5,')  =  ',F10.5,7X,'EPS(',I5,')  =  ',F10.5,
     $7X,'LBD(',I5,')  =  ',F10.5,7X,'UBD(',I5,')  =  ',F10.5)
  183 FORMAT('-',25X,'RIGHT-HAND SIDE VALUES FOR OBJECTIVE FUNCTIONS---R
     $HS')
  184 FORMAT(' ',25X,'RHS(',I5,')  =  ',F20.4)
  185 FORMAT('-',25X,'VECTOR OF ACHIEVEMENT FUNCTION TOLERANCES---EPSY')
  186 FORMAT(' ',25X,'EPSY(',I2,')  =  ',F10.6)
  187 FORMAT('-',25X,'ALL DATA HAS BEEN INPUT---BEGIN THE PATTERN SEARCH
```

```
     $')
  188 FORMAT(' ',25X,'NOTE: IPRINT = 0, SO ONLY THE FINAL SOLUTION WILL
     $BE PRINTED OUT')
  189 FORMAT(' ',25X,'NOTE: IPRINT = 1, SO THE SOLUTION WILL BE PRINTED
     $OUT AT EVERY PATTERN SEARCH CYCLE')
      RETURN
      END
C ***************************************************************************
      SUBROUTINE HJALG
C ***************************************************************************
C
C        SUBROUTINE HJALG IS THE HOOKE-JEEVES PATTERN
C        SEARCH ALGORITHM.  IT PERFORMS EXPLORATORY AND
C        PATTERN MOVES TO FIND THE 'BEST' SOLUTION TO THE
C        NONLINEAR GOAL PROGRAMMING PROBLEM.
C
C
      REAL*4 IPS,LED
      INTEGER*2 IMPROV,IPRINT,ITERM,IPROG,IACCEL
      COMMON/COMM01/NOBJ,NPRIOR,NVAR,NMAX,NCYCLE
      COMMON/COMM02/X(2500),XBASE(2500)
      COMMON/COMM03/A(10),ABEST(10)
      COMMON/COMM04/ACCEL,REDUCE,DELTA,IMPROV,IPRINT,ITERM
      COMMON/COMM05/LBD(2500),UBD(2500),EPS(2500),IPS(2500)
      COMMON/COMM09/SAVE
      CALL ACHFUN
C        SUBROUTINE ACHFUN FORMS THE ACHIEVEMENT VECTOR FOR
C        THE INITIAL BASE POINT.  IT IS TESTED FOR ALL ZERO
C        VALUES---IF SO, THE INITIAL POINT IS OPTIMAL.
C        OTHERWISE, IT IS SET AS THE BEST VALUE AND IS PRINTED.
      DO 100 J=1,NPRIOR
      IF(A(J) .NE. 0.0)GO TO 203
  100 CONTINUE
C        ALL PRIORITY LEVELS ARE SATISFIED AT ZERO VALUE.
C        THE INITIAL POINT IS OPTIMAL---TERMINATE.
  200 PRINT 290
      DO 201 J=1,NPRICE
  201 PRINT 282,J,A(J)
      PRINT 291
```

```
      DO 202 K=1,NVAR
  202 PRINT 280,K,X(K)
      GO TO 295
  203 PRINT 281
      DO 204 J=1,NPRIOR
      ABEST(J) = A(J)
  204 PRINT 282,J,A(J)
C        ** INITIALIZATION OF PATTERN SEARCH PARAMETERS **
C        NCYCLE COUNTS THE NUMBER OF PATTERN SEARCHES USED
C        IACCEL IS AN INDICATOR WHICH TELLS IF THE EXPLORATORY
C        MOVES ARE MADE ABOUT AN ACCELERATION POINT
      NCYCLE = 0
      IACCEL = 0
C        SET THE INITIAL BASE POINT AND THE INITIAL STEP
C        SIZE BEFORE THE EXPLORATORY MOVES
      DO 205 K=1,NVAR
      XBASE(K) = X(K)
  205 IPS(K) = EPS(K)
C        START THE PATTERN SEARCH
C        ** EXPLORATORY MOVES **
  206 IF(NCYCLE .GT. NMAX) GO TO 260
      IPROG = 0
C        IPROG COUNTS THE NUMBER OF IMPROVEMENTS DURING THE
C        EXPLORATORY MOVES.  IT IS USED TO DECIDE IF A PATTERN
C        MOVE SHOULD BE PERFORMED.
  207 DO 210 K=1,NVAR
C        PERTURBATIONS IN THE POSITIVE DIRECTION
C        TEST STEP SIZE COMPONENT.  IF LESS THAN THE MINIMUM
C        ALLOWABLE STEP SIZE, A RIDGE SEARCH WILL BE PERFORMED
      IF(IPS(K) .LT. DELTA) GO TO 255
      X(K) = X(K) + IPS(K)
      IF(X(K) .LT. LBD(K))GO TO 208
      IF(X(K) .GT. UBD(K)) GO TO 208
      CALL ACHFUN
C        ACHFUN FORMS THE ACHIEVEMENT VECTOR AT THE TEST POINT
      ITERM = 0
      CALL DECIDE
C        DECIDE COMPARES THE ACHIEVEMENT VECTORS OF THE
C        BASE POINT AND THE TEST POINT.  IF X(K) IMPROVES THE
C        SOLUTION, IT BECOMES A TEMPORARY HEAD POINT AND
C        EXPLORATORY MOVES ARE CONTINUED FROM IT.  IF THERE
C        IS NO IMPROVEMENT, THEN EXPLORE IN THE NEGATIVE DIRECTION
      IF(IMPROV .EQ. 0) GO TO 208
      IPROG = IPROG + 1
      GO TO 210
C        PERTURBATIONS IN THE NEGATIVE DIRECTION
  208 X(K) = X(K) - 2.0*IPS(K)
      IF(X(K) .LT. LBD(K)) GO TO 209
      IF(X(K) .GT. UBD(K)) GO TO 209
      CALL ACHFUN
C        ACHFUN FORMS THE ACHIEVEMENT VECTOR AT THE TEST POINT
      ITERM = 0
      CALL DECIDE
C        IF X(K) IMPROVES THE SOLUTION, IT BECOMES THE
C        TEMPORARY HEAD POINT AND THE NEXT VARIABLE IS
C        PERTURBED.  IF THERE IS NO IMPROVEMENT, XBASE(K)
C        REMAINS THE TEMPORARY HEAD POINT.
      IF(IMPROV .EQ.0) GO TO 209
      IPROG = IPROG + 1
      GO TO 210
```

```
      IF(IPRINT .EQ. 0)GO TO 206
      PRINT 287,NCYCLE
      DO 241 K=1,NVAR
  241 PRINT 280,K,X(K)
      PRINT 288
      DO 245 J=1,NPRIOR
  245 PRINT 282,J,A(J)
      GO TO 206
C        EXPLORATION AROUND THE ACCELERATION POINT
C        DOES NOT YIELD ANY IMPROVEMENT
C        RESET THE BASE POINT AND GO TO EXPLORATORY MOVES.
  250 IF(IACCEL .EQ. 0) GO TO 255
  251 DO 252 K=1,NVAR
  252 X(K) = XBASE(K)
      IACCEL = 0
      GO TO 206
C        STEP SIZE REDUCTIONS EQUALLED.
C        A RESOLUTION RIDGE SEARCH WILL BE PERFORMED
  255 PRINT 284
  259 CALL RIDGE
C        SUBROUTINE RIDGE WILL TRY TO FIND A NEW BASE POINT
C        IN AN OBLIQUE DIRECTION TO THE PRESENT EXPLORATORY
C        SEARCH DIRECTION.  IF SUCCESSFUL, THE PATTERN
C        SEARCH WILL BEGIN AT THIS NEW POINT.  IF UNSUCCESSFUL,
C        THE ALGORITHM WILL BE TERMINATED.
      IACCEL = 0
      GO TO 206
C        THE MAXIMUM NUMBER OF PATTERN SEARCH CYCLES HAS BEEN
C        EXCEEDED.  THE BEST VALUES OF THE VARIABLES AND THE
C        ACHIEVEMENT VECTOR ARE PRINTED OUT, AND THE ALGORITHM
C        WILL BE TERMINATED.
  260 PRINT 289
      IF(IACCEL .LT.1) GO TO 265
      DO 261 K=1,NVAR
  261 PRINT 280,K,X(K)
      PRINT 288
      DO 262 J=1,NPRIOR
  262 PRINT 282,J,A(J)
      GO TO 295
  265 DO 266 K=1,NVAR
  266 PRINT 280,K,XBASE(K)
      PRINT 288
      DO 267 J=1,NPRIOR
  267 PRINT 282,J,ABEST(J)
  270 GO TO 295
  279 FORMAT('-',25X,'THE EXPLORATORY MOVE HAS BEEN SUCCESSFUL',/,' ',25
     $X,'A PATTERN MOVE WILL BE PERFORMED')
  280 FORMAT(' ',34X,'X(',I5,')  =  ',F13.6)
  281 FORMAT(' ',25X,'INITIAL ACHIEVEMENT VECTOR VALUES')
  282 FORMAT(' ',34X,'A(',I2,')  =  ',F13.6)
  283 FORMAT('-',25X,'TEMPORARY HEAD POINT DOES NOT IMPROVE THE SOLUTION
     $',/,' ',25X,'THE STEP SIZES WILL BE REDUCED',/,' ',25X,'NEW EXPLOR
     $ATORY MOVES WILL BE PERFORMED ABOUT XBASE')
  284 FORMAT(' ',25X,'MAXIMUM STEP SIZE REDUCTIONS EXCEEDED FOR THIS EXP
     $LORATION',/,' ',25X,'A RIDGE SEARCH WILL BE PERFORMED')
  285 FORMAT('-',25X,'THE ACCELERATION POINT YIELDS AN IMPROVED SOLUTION
     $',/,' ',25X,'EXPLORATORY MOVES WILL BEGIN AT THIS NEW POINT')
  286 FORMAT('-',25X,'THE ACCELERATION POINT DOES NOT YIELD AN IMPROVED
     $SOLUTION',/,' ',25X,'EXPLORATORY MOVES WILL BEGIN AT THIS POINT')
  287 FORMAT('-',25X,'THE BEST SOLUTION AT PATTERN SEARCH NUMBER ',I3,'
```

```
     $FOLLOWS')
  288 FORMAT(' ',25X,'THE ACHIEVEMENT VECTOR VALUES ARE:')
  289 FORMAT(' ',25X,'MAXIMUM NUMBER OF PATTERN SEARCH CYCLES EXCEEDED',
     $/,' ',25X,'THE ALGORITHM IS TERMINATED',/,' ',25X,'THE BEST SOLUTI
     $ON TO THIS POINT FOLLOWS')
  290 FORMAT('-',20X,43(1H*),//,' ',25X,'OPTIMAL ACHIEVEMENT VECTOR VALUE
     $S',/,' ',20X,43(1H*))
  291 FORMAT('-',20X,43(1H*),//,' ',25X,'OPTIMAL DECISION VARIABLE VALUES
     $',/,' ',20X,43(1H*))
  295 STOP
      END
C *********************************************************************
```

```
      SUBROUTINE DECIDE
C *********************************************************************
C
C        SUBROUTINE DECIDE IS USED FOR TWO PURPOSES:
C           (1) TO COMPARE THE ACHIEVEMENT VECTOR VALUES OF
C        THE BASE POINT AND A TEST POINT DURING THE EXPLORATORY
C        MOVES AND SELECT THE BEST POINT FOR FUTURE MOVES
C           (2) TO TEST AN ACHIEVEMENT VECTOR FOR TERMINATION
C        OF THE ALGORITHM.  THIS CAN OCCUR IN TWO METHODS:
C             ****TERMINATION BY MEANS OF ZERO FOR ALL
C              PRIORITY LEVELS, AND
C             ****TERMINATION BY MEETING SPECIFIED TOLERANCES
C
C        THE PARAMETER ITERM IS AN INDICATOR FOR TERMINATION
C        TESTING.  IF ITERM = 0, ONLY THE COMPARISON TEST IS
C        PERFORMED.  IF ITERM = 1, THE TERMINATION TESTS WILL
C        ALSO BE PERFORMED.
C
C
      INTEGER*2 IMPROV,IPRINT,ITERM
      COMMON/COMM01/NOBJ,NPRIOR,NVAR,NMAX,NCYCLE
      COMMON/COMM02/X(2500),XBASE(2500)
      COMMON/COMM03/A(10),ABEST(10)
      COMMON/COMM04/ACCEL,REDUC,DELTA,IMPROV,IPRINT,ITERM
      COMMON/COMM08/EPSY(10)
C        ** COMPARISON OF ACHIEVEMENT VECTORS FOR EXPLORATORY MOVES **
  600 DO 610 J=1,NPRIOR
      IF(A(J) .GT. ABEST(J) +0.0001) GO TO 630
      IF(A(J) +0.0001 .LT. ABEST(J))GO TO 620
      IF(J .EQ. NPRIOR) GO TO 630
  610 CONTINUE
```

```
C        THE TEST POINT IMPROVES THE SOLUTION
  620 IMPROV = 1
      GO TO 640
C        THE TEST POINT DOES NOT IMPROVE THE SOLUTION
  630 IMPROV = 0
C        CHECK THE INDICATOR FOR FURTHER TERMINATION TESTING
  640 IF(ITERM .EQ. 0)GO TO 670
      CONTINUE
C        ** TERMINATION TESTS **
C        (1)TERMINATION TEST BY MEANS OF ZERO FOR ALL PRIORITIES
  650 DO 655 J=1,NPRIOR
      IF(A(J) .NE. 0.0)GO TO 660
  655 CONTINUE
C        ALL PRIORITY LEVELS ARE ZERO---THE OPTIMAL SOLUTION
C        HAS BEEN FOUND---TERMINATE THE ALGORITHM
      GO TO 680
C        (2)TERMINATION TEST BY TOLERANCES
  660 DO 665 J=1,NPRIOR
      IF(ABS(A(J) - ABEST(J)) .GT. EPSY(J))GO TO 670
  665 CONTINUE
C        ALL TOLERANCES ARE SATISFIED---A SOLUTION HAS BEEN
C        FOUND---THE SEARCH ALGORITHM IS COMPLETE
      GO TO 680
C        A TOLERANCE IS NOT SATISFIED---A BETTER SOLUTION
C        WILL BE SOUGHT---RETURN TO THE PATTERN SEARCH
  670 IF(IMPROV .EQ. 0)GO TO 695
      DO 675 J=1,NPRIOR
  675 ABEST(J) = A(J)
      GO TO 695
  680 PRINT 690
      PRINT 691
      DO 682 J=1,NPRIOR
  682 PRINT 692,J,A(J)
      PRINT 693
      DO 685 K=1,NVAR
  685 PRINT 694,K,X(K)
      STOP
  690 FORMAT('-',25X,'THE SEARCH ALGORITHM IS COMPLETE---ALL TOLERANCES
     $FOR A SOLUTION ARE SATISFIED')
  691 FORMAT('-',20X,43(1H*),/,' ',25X,'OPTIMAL ACHIEVEMENT VECTOR VALUE
     $S',/,' ',20X,43(1H*))
  692 FORMAT(' ',34X,'A(',I2,')  =  ',F13.6)
  693 FORMAT('-',20X,43(1H*),/,' ',25X,'OPTIMAL DECISION VARIABLE VALUES
     $',/,' ',20X,43(1H*))
  694 FORMAT(' ',34X,'X(',I5,')  =  ',F13.6)
  695 RETURN
      END
C ***************************************************************************
```

```
      SUBROUTINE RIDGE
C *****************************************************************************
C
C        SUBROUTINE RIDGE IS CALLED WHEN THE PATTERN SEARCH
C        CAN NO LONGER FIND PATTERN OR EXPLORATORY MOVES
C        WHICH IMPROVE THE ACHIEVEMENT VECTOR.  RIDGE
C        EVALUATES EXPLORATORY POINTS WHICH ARE IN OBLIQUE
C        DIRECTIONS TO THE USUAL EXPLORATION AXES.  RIDGE
C        ATTEMPTS TO FIND A RESOLUTION RIDGE, IF IT EXISTS,
C        AND MOVE THE PATTERN IN THAT DIRECTION.
C
C
      DIMENSION APOS(10),ANEG(10)
      REAL*4 IPS,LBD
      INTEGER*2 IMPROV,IPRINT,ITERM
      COMMON/COMM01/NOBJ,NPRIOR,NVAR,NMAX,NCYCLE
      COMMON/COMM02/X(2500),XBASE(2500)
      COMMON/COMM03/A(10),ABEST(10)
      COMMON/COMM04/ACCEL,REDUCE,DELTA,IMPROV,IPRINT,ITERM
      COMMON/COMM05/LBD(2500),UBD(2500),EPS(2500),IPS(2500)
      COMMON/COMM09/SAVE
C        PERTURB EACH VARIABLE ONE AT A TIME
C        PERTURB IN THE POSITIVE DIRECTION
      DO 800 K=1,NVAR
  300 X(K) = XBASE(K)
  801 DO 865 K=1,NVAR
      X(K) = X(K) + IPS(K)
      IF(X(K) .LT. LBD(K)) GO TO 805
      IF(X(K) .GT. UBD(K)) GO TO 810
      GO TO 815
  805 X(K) = LBD(K)
      GO TO 815
  810 X(K) = UBD(K)
  815 CALL ACHFUN
C        ACHFUN FORMS THE ACHIEVEMENT VECTOR IN THE POSITIVE
C        DIRECTION
      DO 820 J=1,NPRIOR
  820 APOS(J) = A(J)
C        PERTURB IN THE NEGATIVE DIRECTION
      X(K) = X(K) - 2.0 * IPS(K)
      IF(X(K) .LT. LBD(K)) GO TO 825
      IF(X(K) .GT. UBD(K)) GO TO 830
      GO TO 835
  825 X(K) = LBD(K)
      GO TO 835
  830 X(K) = UBD(K)
  835 CALL ACHFUN
C        ACHFUN FORMS THE ACHIEVEMENT VECTOR IN THE NEGATIVE
C        DIRECTION
      DO 840 J=1,NPRIOR
  840 ANEG(J) = A(J)
C        COMPARE APOS(J) AND ANEG(J).  SELECT THE MINIMUM
C        VIA A STEPWISE PREEMPTIVE PROCEDURE.  THIS DETERMINES
C        A DIRECTION  FOR THE RIDGE POINT.
      DO 850 J=1,NPRIOR
      IF(APOS(J) .LT. ANEG(J))GO TO 855
      IF(APOS(J) .GT. ANEG(J))GO TO 860
      IF(J .EQ. NPRIOR)GO TO 855
  850 CONTINUE
  855 X(K) = XBASE(K) + IPS(K)
```

```
      GO TC 861
  860 X(K) = XBASE(K) - IPS(K)
C        RETURN X(K) TC ITS BASE POINT VALUE, XBASE, BEFORE
C        THE PERTURBATION OF THE NEXT VARIABLE
  861 SAVE = XBASE(K)
      XBASE(K) = X(K)
  865 X(K) = SAVE
C        THE NEW POINT, XRIDGE, HAS BEEN FORMED IN AN OBLIQUE
C        DIRECTION TO THE OLD EXPLORATORY MOVES.  ITS
C        ACHIEVEMENT VECTOR IS COMPUTED AND COMPARED TO THE
C        BEST VALUE AT POINT XBASE.
      DO 866 K=1,NVAR
      SAVE = XBASE(K)
      XBASE(K) = X(K)
  866 X(K) = SAVE
      CALL ACHFUN
      ITERM = 0
      CALL DECIDE
C        IF THE RIDGE POINT GIVES NO IMPROVEMENT, THE PRESENT
C        BASE POINT IS THE BEST SOLUTION, AND THE ALGORITHM
C        IS TERMINATED.
C        IF THE RIDGE POINT GIVES IMPROVEMENT, AN EVEN BETTER
C        POINT WILL BE SOUGHT ALONG THE RESOLUTION RIDGE.
      IF(IMPROV .EQ.1) GO TO 869
      CONTINUE
C        XBASE IS THE BEST SOLUTION
      PRINT 891
      PRINT 897
      DO 867 K=1,NVAR
  867 PRINT 892,K,XBASE(K)
      PRINT 898
      DO 868 J=1,NPRIOR
  868 PRINT 893,J,ABEST(J)
      STOP
C        THE RIDGE POINT GIVES AN IMPROVED SOLUTION.
C        TRY TO ACCELERATE ALONG THE RESOLUTION RIDGE
  869 PRINT 894
  870 DO 871 K=1,NVAR
      SAVE = X(K)
      X(K) = X(K) + ACCEL * (X(K) - XBASE(K))
  871 XBASE(K) = SAVE
      CALL ACHFUN
      ITERM = 0
      CALL DECIDE
C        IF THE ACCELERATION POINT YIELDS AN IMPROVED SOLUTION,
C        TRY TO ACCELERATE FARTHER ALONG THE RESOLUTION RIDGE
C        IF THERE IS NO IMPROVEMENT, THE RIDGE POINT BECOMES
C        THE NEW POINT FOR THE RENEWED PATTERN SEARCH.
      IF(IMPROV .EQ. 0) GO TO 872
      PRINT 896
      GO TO 870
  872 PRINT 895
      PRINT 897
      DO 873 K=1,NVAR
  873 PRINT 892,K,XBASE(K)
      PRINT 898
      DO 875 J=1,NPRIOR
  875 PRINT 893,J,ABEST(J)
C        RESET THE BASE POINT AND STEP SIZE VECTORS AND
C        RETURN TO THE EXPLORATORY MOVES.
```

```
C        NO IMPROVEMENT FROM ACCELERATION POINT
      DO 877 K=1,NVAR
      X(K) = XBASE(K)
  877 IPS(K) = EPS(K)
      DO 879 J=1,NPRIOR
  879 A(J) = ABEST(J)
      GO TO 899
  891 FORMAT('-',25X,'THE RIDGE SEARCH IS COMPLETE',/,' ',25X,'NO IMPROV
     $ED SOLUTION CAN BE FOUND',/,' ',25X,'THE CURRENT BASE POINT IS THE
     $ BEST SOLUTION THAT CAN BE FOUND THROUGH THE PATTERN SEARCH')
  892 FORMAT(' ',34X,'X(',I5,') = ',F13.6)
  893 FORMAT(' ',34X,'A(',I2,')     = ',F13.6)
  894 FORMAT('-',25X,'THE RIDGE POINT GIVES AN IMPROVED SOLUTION',/,' ',
     $25X,'TRY TO ACCELERATE ALONG THE RESOLUTION RIDGE')
  895 FORMAT('-',25X,'ACCELERATION ALONG THE RESOLUTION RIDGE YIELDS NO
     $IMPROVED SOLUTION',/,' ',25X,'THE RIDGE POINT IS THE NEW BASE POIN
     $T')
  896 FORMAT('-',25X,'ACCELERATION ALONG THE RESOLUTION RIDGE YIELDS AN
     $IMPROVED SOLUTION',/,' ',25X,'TRY TO ACCELERATE FURTHER')
  897 FORMAT('-',25X,'DECISION VARIABLES')
  898 FORMAT('-',25X,'ACHIEVEMENT VALUES')
  899 RETURN
      END
C *****************************************************************
```

```
      SUBROUTINE DEVVAR
C *****************************************************************
C
C        SUBROUTINE DEVVAR CALCULATES THE POSITIVE AND
C        NEGATIVE DEVIATION VARIABLES FOR EACH OBJECTIVE
C        FUNCTION USING THE PREVIOUSLY EVALUATED OBJECTIVE
C        FUNCTIONS AND THE RIGHT-HAND SIDE VALUES.
C
C
      REAL*4 N
      COMMON/COMM01/NOBJ,NPRIOR,NVAR,NMAX,NCYCLE
      COMMON/COMM02/X(2500),XBASE(2500)
      COMMON/COMM06/OBJ(2500),RHS(2500)
      COMMON/COMM07/N(2500),P(2500)
      CALL OBJFUN
  400 DO 430 I=1,NOBJ
      IF(OBJ(I) - RHS(I))410,420,420
  410 N(I) = RHS(I) - OBJ(I)
      P(I) = 0.0
      GO TO 430
  420 N(I) = 0.00
      P(I) = OBJ(I) - RHS(I)
  430 CONTINUE
      RETURN
      END
C *****************************************************************
```

```
      SUBROUTINE ACHFUN
C ***************************************************************************
C
      REAL*4 N
      COMMON/COMM01/NOBJ,NPRIOR,NVAR,NMAX,NCYCLE
      COMMON/COMM02/X(2500),XBASE(2500)
      COMMON/COMM03/A(10),ABEST(10)
      COMMON/COMM07/N(2500),F(2500)
      CALL DEVVAR
C        ACHIEVEMENT FUNCTIONS ARE ENTERED HERE.
      RETURN
      END
C ***************************************************************************
```

```
      SUBROUTINE OBJFUN
C ***************************************************************************
C
      COMMON/COMM01/NOBJ,NPRIOR,NVAR,NMAX,NCYCLE
      COMMON/COMM02/X(2500),XBASE(2500)
      COMMON/COMM06/OBJ(2500),RHS(2500)
C        OBJECTIVE FUNCTIONS ARE ENTERED HERE
      RETURN
      END
```

APPENDIX C

NLGP/MPS-RS CODE: VERSION 3

Note: Version 3 is set up to solve the Target Allocation Problem.

```
C ***********************************************************************
C       NONLINEAR GOAL PROGRAMMING--PATTERN SEARCH CODE
C ***********************************************************************
C
C            THIS IS THE NLGP/MPS-PS NAVY CODE
C            NONLINEAR GOAL PROGRAMMING/MODIFIED PATTERN SEARCH CODE
C            WITH A RESOLUTION RIDGE SEARCH TECHNIQUE
C            VERSION 3: SUBROUTINE OVALUE
C                 --- JULY 15, 1980 ---
C
C
C                 ***CODE SPECIFICATIONS***
C                    2500 OBJECTIVES
C                    2500 VARIABLES
C                      10 PRIORITY LEVELS
C
C                 ***VARIABLES AND PARAMETERS***
C     ***NAME***          ***DESCRIPTION***
C        NOBJ                NUMBER OF OBJECTIVES
C        NPRIOR              NUMBER OF PRIORITY LEVELS
C        NVAR                NUMBER OF DECISION VARIABLES
C        NMAX                MAXIMUM NO. OF PATTERN SEARCH CYCLES ALLOWED
C        NCYCLE              COUNTER FOR PATTERN SEARCH CYCLES
C        X                   DECISION VARIABLE VECTOR USED FOR PERTURBATION
C        XBASE               BASE POINT VECTOR FOR PATTERN SEARCH
C        A                   ACHIEVEMENT VECTOR AT TEST POINT
C        ABEST               ACHIEVEMENT VECTOR FOR BEST SOLUTION
C        APOS,ANEG           TEMPORARY ACHIEVEMENT VECTORS FOR RIDGE SEACH
C        EPS,IPS             SCALARS OF PERTURBATION STEP SIZES USED FOR
C                            EXPLORATORY MOVES
C        LBD,UBD             LOWER AND UPPER BOUND VECTORS FOR DECISION VAR
C        OBJ                 OBJECTIVE FUNCTION VECTOR
C        RHS                 RIGHT-HAND SIDE VALUE VECTOR FOR OBJECTIVE FNS
C        N,P                 NEGATIVE AND POSITIVE DEVIATION VARIABLE VECT.
C        EPSY                TERMINATION TOLERANCE TEST VECTOR
C        ACCEL               ACCELERATION FACTOR
C        REDUCE              STEP SIZE REDUCTION FACTOR
C        DELTA               MINIMUM ALLOWABLE STEP SIZE
C        IMPROV              TEST SWITCH---IF IMPROV=1, X IMPROVES THE
C                            SOLUTION; IF IMPROV=0, X DOES NOT IMPROVE THE
C                            SOLUTION
C        IPRINT              OUTPUT SWITCH---IF IPRINT=1, RESULTS ARE
C                            PRINTED OUT AT EVERY PATTERN SEARCH CYCLE;
C                            IF IPRINT=0, ONLY THE FINAL RESULTS ARE OUTPUT
C        ITERM               TERMINATION TEST SWITCH---IF ITERM=1, THE
C                            SOLUTION IS TESTED FOR TERMINATION; IF ITERM=0
C                            ONLY A COMPARISON TEST IS PERFORMED
C        IPROG               COUNTER FOR IMPROVEMENTS DURING THE EXPLORA-
C                            TORY MOVES
C        IACCEL              INDICATOR FOR ACCELERATION MOVES:
C                            IACCEL = 1, ACCELERATION MOVE WAS SUCCESSFUL
C                            IACCEL =-1, ACCELERATION MOVE WAS UNSUCCESSFUL
C                            IACCEL = 0, ACCELERATION MOVE WAS NOT ATTEMPTED
C        SAVE                A TEMPORARY SWITCHING INDICATOR FOR PATTERN
C                            AND RIDGE MOVES
C        JPI                 INTERNAL CONTROL PARAMETER FOR SUBROUTINE
C                            OVALUE---IF JPI=0, THEN OVALUE IS NOT USED;
C                            IF JPI = 1, OVALUE IS USED TO EVALUATE THE
C                            OBJECTIVE FUNCTIONS DURING THE EXPLORATORY
C                            PERTURBATIONS
```

```
C         AMAP              MATRIX OF OBJECTIVE FUNCTION COEFFICIENTS
C         DIFF              DIFFERENCES BETWEEN OBJECTIVE FUNCTION VALUES
C                           FOR THE BEST POINT AND THE CURRENT TEST POINT
C         LNUM              INDEX OF VARIABLE BEING PERTURBED AT
C                           EXPLORATORY SEARCH
C         SEPS              DIRECTIONAL STEP SIZE SCALAR-IPS---IF POSITIVE
C                           DIRECTION EXPLORATORY MOVE, SEPS=IPS; IF NEGA-
C                           TIVE DIRECTION EXPLORATORY MOVE, SEPS=-IPS
C
C
C                     ***DATA INPUT GUIDE***
C
C--------CARD 1                              FORMAT
C         NOBJ                              I5
C         NPRICR                            I5
C         NVAR                              I5
C         NMAX                              I5
C         IPRINT                            I5
C--------CARD 2---INITIAL POINT
C         X(K), K=1,NVAR                    8F10.0
C--------CARD 3---LOWER BOUNDS
C         LBD(K), K=1,NVAR                  8F10.0
C--------CARD 4---UPPER BOUNDS
C         UBD(K), K=1,NVAR                  8F10.0
C--------CARD 5---RIGHT-HAND SIDES
C         RHS(I), I=1,NOBJ                  8F10.0
C--------CARD 6---TOLERANCES FOR TERMINATION
C         EPSY(J),J=1,NPRICR                8F10.0
C--------CARD 7
C         EPS---INITIAL STEP SIZE            F10.0
C         ACCEL---ACCELERATION FACTOR        F10.0
C         REDUCE---STEP SIZE REDUCTION
C                  FACTOR                    F10.0
C         DELTA---STEP SIZE TEST FACTOR      F10.0
C--------CARD 8
C         AMAP(K,J),J=1,NVAR FOR J-TH        F10.0
C                  OBJECTIVE FUNCTION
C
C
C                     ***SUBROUTINES***
C         SUBROUTINES OBJFUN,ACHFUN, AND DVALUE MUST BE USER SUPPLIED
C         FOR EACH NEW NLGP PROBLEM
C         SUBROUTINE OBJFUN CONTAINS THE OBJECTIVE FUNCTIONS
C         SUBROUTINE ACHFUN CONTAINS THE ACHIEVEMENT
C         FUNCTIONS, ONE FOR EACH PRIORITY LEVEL
C         SUBROUTINE DVALUE CONTAINS THE OBJECTIVE FUNCTION DIFFERENCES
C
C
C                     ***MAIN PROGRAM***
C
      REAL*4 IPS,LBD,N
      INTEGER*2 IMPROV,IPRINT,ITERM
      COMMON/COMM01/NOBJ,NPRIOR,NVAR,NMAX,NCYCLE
      COMMON/COMM02/X(2500),XBASE(2500)
      COMMON/COMM03/A(10),ABEST(10)
      COMMON/COMM04/ACCEL,REDUCE,DELTA,IMPROV,IPRINT,ITERM
      COMMON/COMM05/LBD(2500),UBD(2500),EPS,IPS
      COMMON/COMM06/OBJ(2500),RHS(2500)
      COMMON/COMM07/N(2500),P(2500)
      COMMON/COMM08/EPSY(10)
```

```
      COMMON/COMM09/SAVE
      COMMON/COMM10/JPI,LNUM
      COMMON/COMM11/DIFF(2500)
      COMMON/COMM12/AMAP(100,100)
      COMMON/COMM13/EPS
C
C        BEGIN THE PROGRAM
      CALL DATAIN
C        ALL NECESSARY DATA IS INPUT AND PRINTED OUT
      CALL HJALG
C        BEGIN THE PATTERN SEARCH ALGORITHM
C        ALL FURTHER WORK IS COMPLETED IN THE SUBROUTINES
C        OF THE CODE
      STOP
      END
C ***************************************************************************
      SUBROUTINE DATAIN
C ***************************************************************************
C
C        SUBROUTINE DATAIN READS IN ALL THE NECESSARY DATA
C        AND PRINTS IT OUT AS A MEANS OF CHECKING IT.
C
C
      INTEGER*2 IMPROV,IPRINT,ITERM
      REAL*4 IPS,LBD
      COMMON/COMM01/NOBJ,NPRIOR,NVAR,NMAX,NCYCLE
      COMMON/COMM02/X(2500),XBASE(2500)
      COMMON/COMM04/ACCEL,REDUCE,DELTA,IMPROV,IPRINT,ITERM
      COMMON/COMM05/LBD(2500),UBD(2500),EPS,IPS
      COMMON/COMM06/OBJ(2500),RHS(2500)
      COMMON/COMM08/EPSY(10)
      COMMON/COMM10/JPI,LNUM
      COMMON/COMM12/AMAP(100,100)
C        READ ALL THE NECESSARY DATA
      READ 170,NOBJ,NPRIOR,NVAR,NMAX,IPRINT
      READ 175,(X(K),K=1,NVAR)
      READ 175,(LBD(K),K=1,NVAR)
      READ 175,(UBD(K),K=1,NVAR)
      READ 175,(RHS(I),I=1,NOBJ)
      READ 175,(EPSY(J),J=1,NPRIOR)
      READ 175,EPS,ACCEL,REDUCE,DELTA
      DO 100 J=1,NOBJ
  100 READ 175,(AMAP(L,J),I=1,NVAR)
C        PRINT AN ECHO CHECK OF THE DATA
  105 PRINT 180,NOBJ,NPRIOR,NVAR,NMAX,EPS,ACCEL,REDUCE,DELTA
      PRINT 181
      DO 110 K=1,NVAR
  110 PRINT 182,K,X(K),K,LBD(K),K,UBD(K)
      PRINT 183
      DO 120 I=1,NOBJ
  120 PRINT 184,I,RHS(I)
      PRINT 185
      DO 130 J=1,NPRIOR
  130 PRINT 186,J,EPSY(J)
      PRINT 187
      IF(IPRINT .EQ. 1) GO TO 140
      PRINT 188
      RETURN
  170 FORMAT(16I5)
  140 PRINT 189
```

```
  175 FORMAT(8F10.0)
  180 FORMAT('-',25X,'NUMBER OF OBJECTIVES =',I57,I5,/,' ',25X,'NUMBER O
     $F PRIORITIES =',I57,I5,/,' ',25X,'NUMBER OF DECISION VARIABLES =',
     $I57,I5,/,' ',25X,'MAXIMUM NUMBER OF ALLOWABLE',/,' ',29X,'PATTERN
     $SEARCH CYCLES =',I57,I5,/,' ',25X,'INITIAL STEP  SIZES =',I59,F10.
     $6,/,' ',25X,'ACCELERATION FACTOR =',I59,F10.6,/,' ',25X,'STEP SIZE
     $REDUCTION FACTOR =',I59,F10.6,/,' ',25X,'MINIMUM ALLOWABLE STEP SI
     $ZE =',I59,F10.6)
  181 FORMAT('-',25X,'VECTOR OF INITIAL ESTIMATES FOR DECISION VARIABLES
     $---X',/,' ',25X,'LOWER AND UPPER BOUNDS ON DECISION VARIABLES---LB
     $D AND UBD')
  182 FORMAT(' ',10X,'X(',I5,')  =  ',F10.5,7X,'LBD(',I5,')  =  ',F10.5,
     $7X,'UBD(',I5,')  =  ',F10.5)
  183 FORMAT('-',25X,'RIGHT-HAND SIDE VALUES FOR OBJECTIVE FUNCTIONS---R
     $HS')
  184 FORMAT(' ',25X,'RHS(',I5,')  =  ',F13.4)
  185 FORMAT('-',25X,'VECTOR OF ACHIEVEMENT FUNCTION TOLERANCES---EPSY')
  186 FORMAT(' ',25X,'EPSY(',I2,')  =  ',F10.6)
  187 FORMAT('-',25X,'ALL DATA HAS BEEN INPUT---BEGIN THE PATTERN SEARCH
     $')
  188 FORMAT(' ',25X,'NOTE: IPRINT = 0, SO ONLY THE FINAL SOLUTION WILL
     $BE PRINTED OUT')
  189 FORMAT(' ',25X,'NOTE: IPRINT = 1, SO THE SOLUTION WILL BE PRINTED
     $OUT AT EVERY PATTERN SEARCH CYCLE')
      RETURN
      END
C *************************************************************************
      SUBROUTINE HJAIG
C *************************************************************************
C
C        SUBROUTINE HJAIG IS THE HOOKE-JEEVES PATTERN
C        SEARCH ALGORITHM.  IT PERFORMS EXPLORATORY AND
C        PATTERN MOVES TO FIND THE 'BEST' SOLUTION TO THE
C        NONLINEAR GOAL PROGRAMMING PROBLEM.
C
C
      REAL*4 IPS,LBD
      INTEGER*2 IMPROV,IPRINT,ITERM
      COMMON/COMM01/NOBJ,NPRIOR,NVAR,MAX,NCYCLE
      COMMON/COMM02/X(2500),XBASE(2500)
      COMMON/COMM03/A(10),ABEST(10)
      COMMON/COMM04/ACCEL,BRIDUCE,DELTA,IMPROV,IPRINT,ITERM
      COMMON/COMM05/LBD(2500),UBD(2500),BES,IES
      COMMON/COMM09/SAVE
      COMMON/COMM10/JPI,LNUM
      COMMON/COMM13/EPS
      JPI=0
      CALL ACHFUN
C        SUBROUTINE ACHFUN FORMS THE ACHIEVEMENT VECTOR FOR
C        THE INITIAL BASE POINT.  IT IS TESTED FOR ALL ZERO
C        VALUES---IF SO, THE INITIAL POINT IS OPTIMAL.
C        OTHERWISE, IT IS SET AS THE BEST VALUE AND IS PRINTED.
      DO 100 J=1,NPRIOR
      IF(A(J) .NE. 0.0) GO TO 200
  100 CONTINUE
C        ALL PRIORITY LEVELS ARE SATISFIED AT ZERO VALUE.
C        THE INITIAL POINT IS OPTIMAL---TERMINATE.
  200 PRINT 290
      DO 201 J=1,NPRIOR
  201 PRINT 202,J,A(J)
```

```
      PRINT 291
      DO 202 K=1,NVAR
  202 PRINT 280,K,X(K)
      GO TO 295
  203 PRINT 281
      DO 204 J=1,NPRIOR
      ABEST(J) = A(J)
  204 PRINT 282,J,A(J)
C        ** INITIALIZATION OF PATTERN SEARCH PARAMETERS **
C        NCYCLE COUNTS THE NUMBER OF PATTERN SEARCHES USED
C        IACCEL IS AN INDICATOR WHICH TELLS IF THE EXPLORATORY
C        MOVES ARE MADE ABOUT AN ACCELERATION POINT
      NCYCLE = 0
      IACCEL = 0
C        SET THE INITIAL BASE POINT AND THE INITIAL STEP
C        SIZE VECTOR BEFORE EXPLORATORY MOVES
      DO 205 K=1,NVAR
  205 XBASE(K) = X(K)
      IPS = EPS
C        START THE PATTERN SEARCH
C        ** EXPLORATORY MOVES **
  206 IF(NCYCLE .GT. NMAX) GO TO 260
      JPI = 1
      IPROG = 0
C        IPROG COUNTS THE NUMBER OF IMPROVEMENTS DURING THE
C        EXPLORATORY MOVES.  IT IS USED TO DECIDE IF A PATTERN
C        MOVE SHOULD BE PERFORMED.
  207 DO 210 K=1,NVAR
C        PERTURBATIONS IN THE POSITIVE DIRECTION
C        TEST STEP SIZE COMPONENT.  IF LESS THAN THE MINIMUM
C        ALLOWABLE STEP SIZE, A RIDGE SEARCH WILL BE PERFORMED
      IF(IPS .LT. DELTA) GO TO 255
      LNUM = K
      X(K) = X(K) + IPS
      SEPS = IPS
      IF(X(K) .LT. LBD(K))GO TO 208
      IF(X(K) .GT. UBD(K)) GO TO 208
      CALL ACHFUN
C        ACHFUN FORMS THE ACHIEVEMENT VECTOR AT THE TEST POINT
      ITERM = 0
      IF(K .EQ. NVAR) ITERM = 1
      CALL DECIDE
C        DECIDE COMPARES THE ACHIEVEMENT VECTORS OF THE
C        BASE POINT AND THE TEST POINT.  IF X(K) IMPROVES THE
C        SOLUTION, IT BECOMES A TEMPORARY BASE POINT AND
C        EXPLORATORY MOVES ARE CONTINUED FROM IT.  IF THERE
C        IS NO IMPROVEMENT, THEN EXPLORE IN THE NEGATIVE DIRECTION
      IF(IMPROV .EQ. 0)  GO TO 208
      IPROG = IPROG + 1
      GO TO 210
C        PERTURBATIONS IN THE NEGATIVE DIRECTION
  208 X(K) = X(K) - 2.0*IPS
      SEPS = -IPS
      IF(X(K) .LT. LBD(K)) GO TO 209
      IF(X(K) .GT. UBD(K)) GO TO 209
      CALL ACHFUN
C        ACHFUN FORMS THE ACHIEVEMENT VECTOR AT THE TEST POINT
      ITERM = 0
      IF(K .EQ. NVAR) ITERM = 1
      CALL DECIDE
```

```
C        IF X(K) IMPROVES THE SOLUTION, IT BECOMES THE
C        TEMPORARY BASE POINT AND THE NEXT VARIABLE IS
C        PERTURBED.  IF THERE IS NO IMPROVEMENT, XBASE(K)
C        REMAINS THE TEMPORARY BASE POINT.
      IF(IMPROV .EQ.0)  GO TO 209
      IPROG = IPROG + 1
      GO TO 210
  209 X(K) = X(K) + IPS
  210 CONTINUE
      NCYCLE = NCYCLE + 1
C        THE EXPLORATORY MOVE DO LOOP HAS CONSTRUCTED A
C        NEW POINT.  TEST THE ACHIEVEMENT VECTOR
C        FOR ALL ZERO VALUES
      DO 211 J=1,NPRIOR
      IF(A(J) .NE. 0.000)  GO TO 212
  211 CONTINUE
      GO TO 200
C        THE ACHIEVEMENT VECTOR IS NOT ALL ZEROES
C        TEST THE POINT FOR IMPROVEMENT.  IF IT GIVES
C        IMPROVEMENT, A PATTERN MOVE IS PERFORMED.  IF NOT
C        THEN STEP SIZES WILL BE REDUCED AND NEW EXPLORATORY
C        MOVES WILL BE PERFORMED
  212 IF(IPROG .GT. 0)  GO TO 214
      IF(IACCEL)250,213,215
C        THE EXPLORATORY MOVE HAS BEEN UNSUCCESSFUL.  STEP
C        SIZES WILL BE REDUCED AND NEW EXPLORATORY MOVES
C        WILL BEGIN.
  213 PRINT 283
C        ** REDUCE STEP SIZES **
      IPS = IPS * REDUCE
      GO TO 206
C        THE EXPLORATORY MOVE HAS BEEN SUCCESSFUL.
C        A PATTERN MOVE WILL BE PERFORMED
  214 PRINT 279
C        ** ACCELERATION **
      JPI = 0
  215 DO 220 K=1,NVAR
      SAVE = X(K)
      X(K) = X(K) + ACCEL * (X(K) - XBASE(K))
      IF(X(K) .LT.LBD(K))X(K) = LBD(K)
      IF(X(K) .LE. UBD(K))GO TO 220
      X(K) = UBD(K)
  220 XBASE(K) = SAVE
C        TEST THE ACHIEVEMENT VECTOR AT THE ACCELERATION POINT
      CALL ACHFUN
      ITERM = 1
      CALL DECIDE
      IF(IMPROV .EQ. 1)  GO TO 225
      IACCEL = -1
      GO TO 230
  225 IACCEL = 1
      GO TO 240
C        THE ACCELERATION PT. GIVES NO IMPROVEMENT.
C        EXPLORATORY MOVES WILL BE PERFORMED ABOUT THE
C        ACCELERATION PT. TO FIND IMPROVEMENT.
  230 PRINT 286
  231 IF(IPRINT .EQ.0) GO TO 206
      PRINT 287,NCYCLE
      DO 232 K=1,NVAR
  232 PRINT 280,K,XBASE(K)
```

```
      PRINT 288
      DO 235 J=1,NPRIOR
  235 PRINT 282,J,ABEST(J)
      GO TO 206
C        THE ACCELERATION POINT GIVES IMPROVEMENT.
C        EXPLORE ABOUT IT FOR IMPROVEMENT.
  240 PRINT 285
      IF(IPRINT .EQ. 0) GO TO 206
      PRINT 287,NCYCLE
      DO 241 K=1,NVAR
  241 PRINT 280,K,X(K)
      PRINT 288
      DO 245 J=1,NPRIOR
  245 PRINT 282,J,A(J)
      GO TO 206
C        EXPLORATION AROUND THE ACCELERATION POINT
C        DOES NOT YIELD ANY IMPROVEMENT
C        RESET THE BASE POINT AND GO TO EXPLORATORY MOVES
  250 IF(IACCEL .EQ. 0) GO TO 255
  251 DO 252 K=1,NVAR
  252 X(K) = XBASE(K)
      IACCEL = 0
      GO TO 206
  255 PRINT 284
  259 CALL RIDGE
C        SUBROUTINE RIDGE WILL TRY TO FIND A NEW BASE POINT
C        IN AN OBLIQUE DIRECTION TO THE PRESENT EXPLORATORY
C        SEARCH DIRECTION.  IF SUCCESSFUL, THE PATTERN
C        SEARCH WILL BEGIN AT THIS NEW POINT.  IF UNSUCCESSFUL,
C        THE ALGORITHM WILL BE TERMINATED.
      IACCEL = 0
      GO TO 206
C        THE MAXIMUM NUMBER OF PATTERN SEARCH CYCLES HAS BEEN
C        EXCEEDED.  THE BEST VALUES OF THE VARIABLES AND THE
C        ACHIEVEMENT VECTOR ARE PRINTED OUT, AND THE ALGORITHM
C        WILL BE TERMINATED.
  260 PRINT 289
      IF(IACCEL .LT. 1)  GO TO 265
      DO 261 K=1,NVAR
  261 PRINT 280,K,X(K)
      PRINT 288
      DO 262 J=1,NPRIOR
  262 PRINT 282,J,A(J)
      GO TO 295
  265 DO 266 K=1,NVAR
  266 PRINT 280,K,XBASE(K)
      PRINT 288
      DO 267 J=1,NPRIOR
  267 PRINT 282,J,ABEST(J)
  270 GO TO 295
  279 FORMAT('-',25X,'THE EXPLORATORY MOVE HAS BEEN SUCCESSFUL',/,' ',25
     $X,'A PATTERN MOVE WILL BE PERFORMED')
  280 FORMAT(' ',34X,'X(',I5,')  =  ',F13.6)
  281 FORMAT(' ',25X,'INITIAL ACHIEVEMENT VECTOR VALUES')
  282 FORMAT(' ',34X,'A(',I2,')  =  ',F13.6)
  283 FORMAT('-',25X,'TEMPORARY HEAD POINT DOES NOT IMPROVE THE SOLUTION
     $',/,' ',25X,'THE STEP SIZES WILL BE REDUCED',/,' ',25X,'NEW EXPLOR
     $ATORY MOVES WILL BE PERFORMED ABOUT XBASE')
  284 FORMAT(' ',25X,'MAXIMUM STEP SIZE REDUCTIONS EXCEEDED FOR THIS EXP
     $LORATION',/,' ',25X,'A RIDGE SEARCH WILL BE PERFORMED')
```

```
  285 FORMAT('-',25X,'THE ACCELERATION POINT YIELDS AN IMPROVED SOLUTION
     $',/,' ',25X,'EXPLORATORY MOVES WILL BEGIN AT THIS NEW POINT')
  286 FORMAT('-',25X,'THE ACCELERATION POINT DOES NOT YIELD AN IMPROVED
     $SOLUTION',/,' ',25X,'EXPLORATORY MOVES WILL BEGIN AT THIS POINT')
  287 FORMAT('-',25X,'THE BEST SOLUTION AT PATTERN SEARCH NUMBER ',I3,'
     $FOLLOWS')
  288 FORMAT(' ',25X,'THE ACHIEVEMENT VECTOR VALUES ARE:')
  289 FORMAT(' ',25X,'MAXIMUM NUMBER OF PATTERN SEARCH CYCLES EXCEEDED',
     $/,' ',25X,'THE ALGORITHM IS TERMINATED',/,' ',25X,'THE BEST SOLUTI
     $ON TO THIS POINT FOLLOWS')
  290 FORMAT('-',20X,43(1H*),/,' ',25X,'OPTIMAL ACHIEVEMENT VECTOR VALUE
     $S',/,' ',20X,43(1H*))
  291 FORMAT('-',20X,43(1H*),/,' ',25X,'OPTIMAL DECISION VARIABLE VALUES
     $',/,' ',20X,43(1H*))
  295 STOP
      END
C ***************************************************************
      SUBROUTINE DEVVAR
C ***************************************************************
C
C        SUBROUTINE DEVVAR CALCULATES THE POSITIVE AND
C        NEGATIVE DEVIATION VARIABLES FOR EACH OBJECTIVE
C        FUNCTION USING THE PREVIOUSLY EVALUATED OBJECTIVE
C        FUNCTIONS AND THE RIGHT-HAND SIDE VALUES.
C
C
      REAL*4 N
      COMMON/COMM01/NOBJ,NPRIOR,NVAR,NMAX,NCYCLE
      COMMON/COMM02/X(2500),XBASE(2500)
      COMMON/COMM06/OBJ(2500),RHS(2500)
      COMMON/COMM07/N(2500),P(2500)
      COMMON/COMM10/JPI,LNDX
      IF(JPI.EQ.0) GO TO 400
      CALL LVALUE
      GO TO 405
  400 CALL OBJFUN
  405 DO 430 I=1,NOBJ
      IF(OBJ(I) - RHS(I))410,420,420
  410 N(I) = RHS(I) - OBJ(I)
      P(I) = 0.0
      GO TO 430
  420 N(I) = 0.00
      P(I) = OBJ(I) - RHS(I)
  430 CONTINUE
      RETURN
      END
C ***************************************************************
      SUBROUTINE DECIDE
C ***************************************************************
C
C        SUBROUTINE DECIDE IS USED FOR TWO PURPOSES:
C           (1) TO COMPARE THE ACHIEVEMENT VECTOR VALUES OF
C        THE BASE POINT AND A TEST POINT DURING THE EXPLORATORY
C        MOVES AND SELECT THE BEST POINT FOR FUTURE MOVES
C           (2) TO TEST AN ACHIEVEMENT VECTOR FOR TERMINATION
C        OF THE ALGORITHM.  THIS CAN OCCUR IN TWO METHODS:
C             ****TERMINATION BY MEANS OF ZERO FOR ALL
C              PRIORITY LEVELS, AND
C             ****TERMINATION BY MEETING SPECIFIED TOLERANCES
C
```

```
C         THE PARAMETER ITERM IS AN INDICATOR FOR TERMINATION
C         TESTING.  IF ITERM = 0, ONLY THE COMPARISON TEST IS
C         PERFORMED.  IF ITERM = 1, THE TERMINATION TESTS WILL
C         ALSO BE PERFORMED.
C
C
      INTEGER*2 IMPROV,IPRINT,ITERM
      COMMON/COMM01/NOBJ,NPRIOR,NVAR,NMAX,NCYCLE
      COMMON/COMM02/X(2500),XBASE(2500)
      COMMON/COMM03/A(10),ABEST(10)
      COMMON/COMM04/ACCEL,REDUCE,DELTA,IMPROV,IPRINT,ITERM
      COMMON/COMM06/OBJ(2500),RHS(2500)
      COMMON/COMM08/EPSY(10)
      COMMON/COMM10/JPI,INUM
      COMMON/COMM11/DIFF(2500)
C        ** COMPARISON OF ACHIEVEMENT VECTORS FOR EXPLORATORY MOVES **
  600 DO 610 J=1,NPRIOR
      IF(A(J) .GT. ABEST(J)+0.00001) GO TO 630
      IF(A(J)+0.00001 .LT. ABEST(J)) GO TO 620
  610 CONTINUE
C     GO TO 630
C        THE TEST POINT IMPROVES THE SOLUTION
  620 IMPROV = 1
C        CHECK THE INDICATOR FOR TERMINATION TESTING
  640 IF(ITERM .GE. 1) GO TO 650
      GO TO 670
C        THE TEST POINT DOES NOT IMPROVE THE SOLUTION
  630 IMPROV = 0
      IF(JPI.EQ.0) GO TO 640
      DO 635 J=1,NOBJ
  635 OBJ(J) = OBJ(J) - DIFF(J)
  640 IF(ITERM .EQ. 2) GO TO 650
      RETURN
C        ** TERMINATION TESTS **
C        (1)TERMINATION TEST BY MEANS OF ZERO FOR ALL PRIORITIES
  650 DO 655 J=1,NPRIOR
      IF(A(J) .NE. 0.0)GO TO 660
  655 CONTINUE
C        ALL PRIORITY LEVELS ARE ZERO---THE OPTIMAL SOLUTION
C        HAS BEEN FOUND---TERMINATE THE ALGORITHM
      GO TO 680
C        (2)TERMINATION TEST BY TOLERANCES
  660 IF(ITERM .EQ. 1) GO TO 670
      DO 665 J=1,NPRIOR
      IF(ABS(A(J) - ABEST(J)) .GT. EPSY(J))GO TO 670
  665 CONTINUE
C        ALL TOLERANCES ARE SATISFIED---A SOLUTION HAS BEEN
C        FOUND---THE SEARCH ALGORITHM IS COMPLETE
      GO TO 690
C        A TOLERANCE IS NOT SATISFIED---A BETTER SOLUTION
C        WILL BE SOUGHT---RETURN TO THE PATTERN SEARCH
  670 IF(IMPROV .EQ. 0) GO TO 695
      DO 675 J=1,NPRIOR
  675 ABEST(J) = A(J)
      RETURN
  680 PRINT 690
      PRINT 691
      DO 682 J=1,NPRIOR
  682 PRINT 692,J,A(J)
      PRINT 693
```

```
      DO 685 K=1,NVAR
  685 PRINT 694,K,X(K)
      STOP
  690 FORMAT('-',25X,'THE SEARCH ALGORITHM IS COMPLETE---ALL TOLERANCES
     $FOR A SOLUTION ARE SATISFIED')
  691 FORMAT('-',20X,43(1H*),/,' ',25X,'OPTIMAL ACHIEVEMENT VECTOR VALUE
     $S',/,' ',20X,43(1H*))
  692 FORMAT(' ',34X,'A(',I2,')  =  ',F13.6)
  693 FORMAT('-',20X,43(1H*),/,' ',25X,'OPTIMAL DECISION VARIABLE VALUES
     $',/,' ',20X,43(1H*))
  694 FORMAT(' ',34X,'X(',I5,')  =  ',F13.6)
  695 RETURN
      END
C *********************************************************************
      SUBROUTINE RIDGE
C *********************************************************************
C
C
C        SUBROUTINE RIDGE IS CALLED WHEN THE PATTERN SEARCH
C        CAN NO LONGER FIND PATTERN OR EXPLORATORY MOVES
C        WHICH IMPROVE THE ACHIEVEMENT VECTOR.  RIDGE
C        EVALUATES EXPLORATORY POINTS WHICH ARE IN OBLIQUE
C        DIRECTIONS TO THE USUAL EXPLORATION AXES.  RIDGE
C        ATTEMPTS TO FIND A RESOLUTION RIDGE, IF IT EXISTS,
C        AND MOVE THE PATTERN IN THAT DIRECTION.
C
C
      DIMENSION APOS(10),ANEG(10)
      REAL*4 IPS,LBD
      INTEGER*2 IMPROV,IPRINT,ITERM
      COMMON/COMM01/NOBJ,NPRIOR,NVAR,NMAX,NCYCLE
      COMMON/COMM02/X(2500),XBASE(2500)
      COMMON/COMM03/A(10),ABEST(10)
      COMMON/COMM04/ACCEL,REDUCE,DELTA,IMPROV,IPRINT,ITERM
      COMMON/COMM05/LBD(2500),UBD(2500),EPS,IPS
      COMMON/COMM09/SAVE
      COMMON/COMM10/JPI,LNUM
      COMMON/COMM13/SEPS
C        PERTURB EACH VARIABLE ONE AT A TIME
C        PERTURB IN THE POSITIVE DIRECTION
      DO 800 K=1,NVAR
  800 X(K) = XBASE(K)
      JPI = 1
  801 DO 865 K=1,NVAR
      X(K) = X(K) + IPS
      IF(X(K) .LT. LBD(K)) GO TO 805
      IF(X(K) .GT. UBD(K)) GO TO 810
      GO TO 815
  805 X(K) = LBD(K)
      GO TO 815
  810 X(K) = UBD(K)
  815 CALL ACHFUN
C        ACHFUN FORMS THE ACHIEVEMENT VECTOR IN THE POSITIVE
C        DIRECTION
      DO 820 J=1,NPRIOR
  820 APOS(J) = A(J)
C        PERTURB IN THE NEGATIVE DIRECTION
      X(K) = X(K) - 2.0 * IPS
      IF(X(K) .LT. LBD(K)) GO TO 825
      IF(X(K) .GT. UBD(K)) GO TO 830
```

```
      GC TC 835
  825 X(K) = LBD(K)
      GC TC 835
  830 X(K) = UBD(K)
  835 CALL ACHFUN
C        ACHFUN FORMS THE ACHIEVEMENT VECTOR IN THE NEGATIVE
C        DIRECTION
      DO 840 J=1,NPRIOR
  840 ANEG(J) = A(J)
C        COMPARE APOS(J) AND ANEG(J).  SELECT THE MINIMUM
C        VIA A STEPWISE PREEMPTIVE PROCEDURE.  THIS DETERMINES
C        A DIRECTION FOR THE RIDGE POINT.
      DO 850 J=1,NPRIOR
      IF(APOS(J) .LT. ANEG(J))GO TO 855
      IF(APOS(J) .GT. ANEG(J))GO TO 860
      IF(J .EQ. NPRIOR)GO TO 855
  850 CONTINUE
  855 X(K) = XBASE(K) + TPS
      GO TO 861
  860 X(K) = XBASE(K) - TPS
C        RETURN X(K) TO ITS BASE POINT VALUE, XBASE, BEFORE
C        THE PERTURBATION OF THE NEXT VARIABLE
  861 SAVE = XBASE(K)
      XBASE(K) = X(K)
  865 X(K) = SAVE
C        THE NEW POINT, XRIDGE, HAS BEEN FORMED IN AN OBLIQUE
C        DIRECTION TO THE OLD EXPLORATORY MOVES.  ITS
C        ACHIEVEMENT VECTOR IS COMPUTED AND COMPARED TO THE
C        BEST VALUE AT POINT XBASE.
      JPI = 0
      DO 866 K=1,NVAR
      SAVE = XBASE(K)
      XBASE(K) = X(K)
  866 X(K) = SAVE
      CALL ACHFUN
      ITERM = 0
      CALL DECIDE
C        IF THE RIDGE POINT GIVES NO IMPROVEMENT, THE PRESENT
C        BASE POINT IS THE BEST SOLUTION, AND THE ALGORITHM
C        IS TERMINATED.
C        IF THE RIDGE POINT GIVES IMPROVEMENT, AN EVEN BETTER
C        POINT WILL BE SOUGHT ALONG THE RESOLUTION RIDGE.
      IF(IMPROV .EQ. 1) GO TO 869
      CONTINUE
C        XBASE IS THE BEST SOLUTION
      PRINT 891
      PRINT 897
      DO 867 K=1,NVAR
  867 PRINT 892,K,XBASE(K)
      PRINT 898
      DO 868 J=1,NPRIOR
  868 PRINT 893,J,ABEST(J)
      STOP
C        THE RIDGE POINT GIVES AN IMPROVED SOLUTION.
C        TRY TO ACCELERATE ALONG THE RESOLUTION RIDGE
  869 PRINT 894
      RACCEL = 2.0 * ACCEL
  870 DO 871 K=1,NVAR
      SAVE = X(K)
      X(K) = X(K) + RACCEL * (X(K) - XBASE(K))
```

```
  871 XBASE(K) = SAVE
      CALL ACHFUN
      ITERM = 0
      CALL IDECIDE
C        IF THE ACCELERATION POINT YIELDS AN IMPROVED SOLUTION,
C        TRY TO ACCELERATE FURTHER ALONG THE RESOLUTION RIDGE
C        IF THERE IS NO IMPROVEMENT, THE RIDGE POINT BECOMES
C        THE NEW POINT FOR THE RENEWED PATTERN SEARCH.
      IF(IMPROV .EQ. 0) GO TO 872
      PRINT 896
      GO TO 870
  872 PRINT 895
      PRINT 897
      DO 873 K=1,NVAR
  873 PRINT 892,K,XBASE(K)
      PRINT 898
      DO 875 J=1,NPRIOR
  875 PRINT 893,J,ABEST(J)
C        RESET THE BASE POINT AND STEP SIZE VECTORS AND
C        RETURN TO THE EXPLORATORY MOVES.
C        NO IMPROVEMENT FROM ACCELERATION POINT
      DO 877 K=1,NVAR
  877 X(K) = XBASE(K)
      IFS = FPS
      DO 879 J=1,NPRIOR
  879 A(J) = ABEST(J)
      GO TO 899
  891 FORMAT('-',25X,'THE RIDGE SEARCH IS COMPLETE',/,' ',25X,'NO IMPROV
     $ED SOLUTION CAN BE FOUND',/,' ',25X,'THE CURRENT BASE POINT IS THE
     $ BEST SOLUTION THAT CAN BE FOUND THROUGH THE PATTERN SEARCH')
  892 FORMAT(' ',34X,'X(',I5,') = ',F13.6)
  893 FORMAT(' ',34X,'A(',I2,')     = ',F13.6)
  894 FORMAT('-',25X,'THE RIDGE POINT GIVES AN IMPROVED SOLUTION',/,' ',
     $25X,'TRY TO ACCELERATE ALONG THE RESOLUTION RIDGE')
  895 FORMAT('-',25X,'ACCELERATION ALONG THE RESOLUTION RIDGE YIELDS NO
     $IMPROVED SOLUTION',/,' ',25X,'THE RIDGE POINT IS THE NEW BASE POIN
     $T')
  896 FORMAT('-',25X,'ACCELERATION ALONG THE RESOLUTION RIDGE YIELDS AN
     $IMPROVED SOLUTION',/,' ',25X,'TRY TO ACCELERATE FURTHER')
  897 FORMAT('-',25X,'DECISION VARIABLES')
  898 FORMAT('-',25X,'ACHIEVEMENT VALUES')
  899 RETURN
      END
C*************************************************************************
      SUBROUTINE OBJFUN
C*************************************************************************
C
C        SUBROUTINE OBJFUN EVALUATES EACH OBJECTIVE
C        FUNCTION.  THIS SUBROUTINE IS PROBLEM DEPENDENT
C        AND MUST BE USER SUPPLIED FOR EACH NLGP PROBLEM.
C
C
      COMMON/COMM02/X(2500),XBASE(2500)
      COMMON/COMM06/OBJ(2500),RHS(2500)
      COMMON/COMM14/A,B,C,D,E
      OBJ(1)=X(1)+X(3)
      OBJ(2)=X(2)+X(4)
      OBJ(3)=2920.*X(1)+1770.*X(2)
      OBJ(4)=2920.*X(3)+1770.*X(4)
      OBJ(5)=X(1)-0.06452*(X(5)+X(7)+X(9))-0.06250*(X(6)+X(8))
```

```
      OBJ(6)=X(2)-0.05556*(X(10)+X(12)+X(14))-0.05264*(X(11)+X(13))
      OBJ(7)=X(3)-0.05896*X(15)-0.06452*(X(16)+X(19))-0.06250*(X(17)+X(1
     58))
      OBJ(8)=X(4)-0.05882*X(20)-0.05556*(X(21)+X(24))-0.05264*(X(22)+X(2
     33))
      A=40.*(0.99978**(X(5)+X(15))*0.99953**(X(10)+X(20)))
      B=10.*(0.99978**(X(6)+X(16))*0.99953**(X(11)+X(21)))
      C=50.*(0.99978**(X(7)+X(17))*0.99953**(X(12)+X(22)))
      D=0.
      E=0.
      OBJ(9) = 100.0 - (A+B+C+D+E)
      RETURN
      END
C*********************************************************************
      SUBROUTINE UVALUE
C*********************************************************************
C
C        SUBROUTINE UVALUE RECALCULATES EACH OBJECTIVE WITHOUT
C        UNNECESSARY EVALUATIONS.  THIS SUBROUTINE IS PROBLEM
C        DEPENDENT.  IF IU = 1 IS DEFINED, USER MUST SUPPLY THIS
C        SUBROUTINE FOR EACH NLP PROBLEM.
C
C
      COMMON/COMM01/NOBJ,NPRIOB,NVAR,NMAX,NCYCLE
      COMMON/COMM02/X(2500),XBASE(2500)
      COMMON/COMM06/OBJ(2500),PRS(2500)
      COMMON/COMM10/JPI,LNUM
      COMMON/COMM11/DIFF(2500)
      COMMON/COMM12/AMAP(100,100)
      COMMON/COMM13/SEPS
      COMMON/COMM14/A,B,C,D,E
      L = LNUM
      DO 999 J=1,NOBJ
      DIFF(J) = 0.0
      IF (AMAP(L,J).EQ.0.0) GO TO 999
      GO TO (101,101,101,101,101,101,101,101,109),J
101   GO TO 1
109   GO TO (999,999,999,999,11,12,13,14,15,11,12,13,14,15,11,12,13,14,1
     55,11,12,13,14,15),L
1     DIFF(J)=AMAP(L,J)*SEPS
      GO TO 998
11    DIFF(J)=SEPS*ALOG(AMAP(L,J))
      A=100.0-B-C-D-E-OBJ(J)
      Y9=ALOG(A)+DIFF(J)
      DIFF(J)=EXP(Y9)-A
      GO TO 998
12    DIFF(J)=SEPS*ALOG(AMAP(L,J))
      E=100.0-A-C-D-E-OBJ(J)
      Y9=ALOG(B)+DIFF(J)
      DIFF(J)=EXP(Y9)-B
      GO TO 998
13    DIFF(J)=SEPS*ALOG(AMAP(L,J))
      C=100.0-A-B-D-E-OBJ(J)
      Y9=ALOG(C)+DIFF(J)
      DIFF(J)=EXP(Y9)-C
      GO TO 998
14    GO TO 999
15    GO TO 999
998   OBJ(J)=OBJ(J)+DIFF(J)
999   CONTINUE
```

```
      RETURN
      END
C *****************************************************************
      SUBROUTINE ACHFUN
C *****************************************************************
C
C         SUBROUTINE ACHFUN FORMS THE ACHIEVEMENT VECTOR
C         FROM THE PREVIOUSLY CALCULATED VALUES OF OBJFUN
C         AND DEVVAR.  SUBROUTINE OBJFUN RETURNS THE VALUES
C         OF THE OBJECTIVE FUNCTIONS AT THE CURRENT TEST PT.
C         SUBROUTINE DEVVAR RETURNS THE VALUES OF THE
C         DEVIATION VARIABLES AT THE CURRENT TEST POINT.
C         THIS SUBROUTINE IS PROBLEM DEPENDENT AND MUST BE
C         USER SUPPLIED FOR EACH NLGP PROBLEM.
C
C
      REAL*4 N
      COMMON/COMM02/X(2500),XBASE(2500)
      COMMON/COMM03/A(10),ABEST(10)
      COMMON/COMM07/N(2500),P(2500)
      CALL DEVVAR
      A(1) = P(1)+P(2)+P(3)+P(4)+N(5)+N(6)+N(7)+N(8)
      A(2) = N(9)
      RETURN
      END
```

BIBLIOGRAPHY

1. Charnes, A. and Cooper, N.W., "Note on an Application of a Goal Programming Model for Media Planning," Management Science, Vol. 14, No. 8, April, 1968.

2. Draus, S., The Design of Acoustic Transducer Arrays Using Goal Programming, M.S. Thesis, The Pennsylvania State University, 1977.

3. Draus, S., Ignizio, J.P., and Wilson, G.L., "The Design of Optimum Sonar Transducer Arrays Using Goal Programming," Proceedings of the 93rd Meeting of the Acoustical Society of America, University Park, PA., June, 1977.

4. Fiacco, A.V. and McCormick, G.P., "Computational Algorithm for the Sequential Unconstrained Minimization Technique for Non-linear Programming," Management Science, Vol. 10, No. 4, 1964.

5. Fiacco, A.V. and McCormick, G.P., "The Sequential Unconstrained Minimization Technique for Non-linear Programming, a Primal-Dual Method," Management Science, Vol. 10, No. 2, 1964.

6. Fletcher, R. and Powell, M.J.D., "A Rapidly Convergent Descent Method for Minimization," Computer Journal, Vol. 6, No. 2, 1963.

7. Glass, H. and Cooper, L., "Sequential Search: A Method for Solving Constrained Optimization Problems," Journal of the Association for Computer Machines, Vol. 12, 1965.

8. Griffith, R.E. and Stewart, R.A., "A Nonlinear Programming Technique for the Optimization of Continuous Processing Systems," Management Science, Vol. 7, 1961.

9. Harkins, A., "The Use of Parallel Tangents in Optimization," Optimization Techniques, 1964.

10. Hestenes, M.R. and Stiefel, E., "Method of Conjugate Gradients for Solving Linear Systems," Journal of Research of the National Bureau of Standards, Vol. 49, 1952.

11. Himsworth, F.R., "Empirical Methods of Optimization," Transactions of the Institute of Chemical Engineers, Vol. 40, 1962.

12. Hooke, R. and Jeeves, T.A., "Direct Search Solution of Numerical and Statistical Problems," Journal of the Association of Computer Machines, Vol. 8, 1961.

13. Ignizio, J.P., "Advances in Multicriteria Engineering Design: Summary Report" (Volume I), for DASG60-77-C-0078, Follow-On, June, 1979.

14. _____ "Advances in Multicriteria Engineering Design: Conformal Arrays" (Volume II), for DASG60-77-C-0078, Follow-On, June, 1979.

15. _____ "Antenna Array Beam Pattern Synthesis Via Goal Programming, European Journal of OR, Vol. 5, 1980.

16. _____ "The Design of Multiple Objective Systems Effectiveness Model for the General Support Rocket Systems," TBE, July, 1975.

17. _____ "A Goal Programming Approach to Antenna Array Design," National ORSA/TIMS Meeting, Las Vegas, Nevada, November, 1975.

18. _____ "Goal Programming and Extensions, Lexington, MA: D.C. Heath and Co., 1976.

19. _____ "Large Scale Multiobjective Systems Synthesis Via Goal Programming," National ORSA/TIMS Meeting, Los Angeles, November 1978.

20. _____ "Multicriteria Optimization in BMD Design," presented at the National ORSA/TIMS Meeting, Atlanta, November, 1977.

21. _____ "Transducer Array Design Using Goal Programming," Report of E/F #6153, for the Naval Research Labs, March, 1976.

22. _____ "The Use of Goal Programming in the Design of Solar Heating and Cooling Systems," TBE, August, 1975.

23. _____ "The Utilization of Goal Programming in Multicriteria Systems Design," National ORSA/TIMS Meeting, Miami, November, 1976.

24. Ignizio, J.P., et al, "Optimization of Multicriteria BMD Problems," Final Report, U.S. Army BMDATC, 1978.

25. Klingman, W.R. and Himmelblau, D.N., "Nonlinear Programming with the Aid of a Multiple-gradient Summation Technique," Journal of the Association for Computer Machines, Vol. 11, No. 4, 1964.

26. Mugele, R.A. "A Nonlinear Digital Optimizing Program for Process Control Systems," Proceedings of the Western Joint Computer Conference, 1962.

27. Powell, M.J.D., "An Efficient Method for Finding the Minimum of a Function of Several Variables without Calculating Derivatives," Computer Journal, Vol. 7, 1964.

28. Rice, E., Bracken, J. and Pennington, A.W., "Allocation of Carrier-based Attack Aircraft Using Nonlinear Programming," Naval Research Logistics Quarterly, Vol. 18, No. 3, 1971.

29. Rosenbrock, H.H., "An Automatic Method for Finding the Greatest or Least Value of a Function," Computer Journal, Vol. 3, No. 3, 1960.

30. Spendley, N., Hext, G.R., and Himsworth, F.R., "Sequential Application of a Simplex Design in Optimization and Evolutionary Operations," Technometrics, Vol. 4, No. 4, 1962.

31. Wilde, D.J., Optimum Seeking Methods, Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1964.

32. Wilde, D.J. and Beighter, C.S., Foundations of Optimization, Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1967.

33. Wood, E.F., "Application of Direct Search to the Solution of Engineering Problems," Westinghouse Scientific Paper 64-1210-1-P, 1960.